SUPPLY CHAIN SCHEDULING: BATCHING AND DELIVERY

NICHOLAS G. HALL

Fisher College of Business, The Ohio State University, Columbus, Ohio 43210, hall_33@cob.osu.edu

CHRIS N. POTTS

Faculty of Mathematical Studies, University of Southampton, Southampton SO17 1BJ, United Kingdom, c.n.potts@maths.soton.ac.uk

Although the supply chain management literature is extensive, the benefits and challenges of coordinated decision making within supply chain *scheduling* models have not been studied. We consider a variety of scheduling, batching, and delivery problems that arise in an arborescent supply chain where a supplier makes deliveries to several manufacturers, who also make deliveries to customers. The objective is to minimize the overall scheduling and delivery cost, using several classical scheduling objectives. This is achieved by scheduling the jobs and forming them into batches, each of which is delivered to the next downstream stage as a single shipment. For each problem, we either derive an efficient dynamic programming algorithm that minimizes the total cost of the supplier or that of the manufacturer, or we demonstrate that this problem is intractable. The total system cost minimization problem of a supplier and manufacturer who make cooperative decisions is also considered. We demonstrate that cooperation between a supplier and a manufacturer may reduce the total system cost by at least 20%, or 25%, or by up to 100%, depending upon the scheduling objective. Finally, we identify incentives and mechanisms for this cooperation, thereby demonstrating that our work has practical implications for improving the efficiency of supply chains.

Received June 2000; revisions received October 2001, August 2002; accepted August 2002.

Subject classifications: Production/scheduling: sequencing, deterministic, single machine, multiple machine. Manufacturing: performance/productivity. Games/group decisions: cooperative, noncooperative.

Area of review: Manufacturing, Service, and Supply Chain Operations.

1. INTRODUCTION

Among the most important and active topics in manufacturing research over the last 10 years has been *supply chain management*. A supply chain represents all the stages at which value is added to a manufactured product, including the supply of raw materials and intermediate components, finished-goods manufacture, packaging, transportation, warehousing, and logistics. Whereas much of the supply chain management literature focuses on inventory control or lot-sizing issues, this paper considers a number of issues that are important to *scheduling* in supply chains. Central to this literature is the idea of *coordination* between different parts of a supply chain. Where decision makers at different stages of a supply chain make decisions that are poorly coordinated, substantial inefficiencies can result. This paper considers the coordination of scheduling, batching, and delivery decisions, both at a single stage and between different stages of a supply chain, to eliminate those inefficiencies. The objective is to minimize the overall scheduling and delivery cost. This is achieved by forming batches of orders, each of which is delivered from a supplier to a manufacturer, or from a manufacturer to a customer, in a single shipment.

Thomas and Griffin (1996) provide an extensive review and discussion of the literature on supply chain management. They emphasize the importance of the problem, in that over 11% of the U.S. Gross National Product is devoted to nonmilitary logistics expenditures. Moreover, for many products, logistics expenditures constitute over 30% of the cost of goods sold. In their conclusions, they discuss the need for research that addresses supply chain issues at an *operational* rather than a strategic level, and that uses *deterministic* rather than stochastic models. This paper addresses both of these needs.

Sarmiento and Nagi (1999) survey the literature of integrated production and distribution models. They motivate the importance of such models by pointing out that the modern trend towards reduced inventory levels creates a closer interaction between adjacent stages in a supply chain. They also discuss several possible topics for future research. Erengüç et al. (1999) provide a similar survey which emphasizes the operational aspects of supply chains and identifies opportunities for using mathematical models. They mention the need for simultaneous decison making at the supplier, plant, and distribution stages of a supply chain, and the related need for effective information sharing.

More similar in spirit to this paper is the literature on production systems with multiple decision makers. Ow et al. (1988) describe a multiagent, or distributed, scheduling system, but do not discuss the benefits of cooperation in detail. Weng (1997) considers a two-stage supply chain in which demand is a function of both price and a random component. He shows that the increase in expected profit resulting from cooperation increases with the price elasticity of demand and the transfer price. Moses and Seshadri (2000) consider a two-stage supply chain with lost sales. They describe an algorithm that finds the optimal fractions in which holding costs should be shared, the review period, and the base-stock level.

We now briefly review the literature on scheduling and batching problems. More extensive surveys can be found in Potts and Van Wassenhove (1992), Webster and Baker (1995), and Potts and Kovalyov (2000). There are two alternative assumptions that may apply for this class of

problems. The first is *batch availability*, under which a job only becomes available for later processing or dispatch to the customer when the entire batch of which it is a part has been processed. Under the *job availability* assumption, however, a job becomes available once it has been processed. This paper uses the batch availability assumption.

Research on problems in which jobs are delivered to customers in batches is limited. Cheng et al. (1996) consider a single-machine batch-scheduling problem, in which they minimize the sum of batch delivery cost plus job-earliness penalties. A similar model, but with given batch delivery dates, is studied by Yang (2000). Lee and Chen (2001) consider the integration of transportation issues, including time and capacity, with scheduling decisions, and study the solvability of a variety of related models. Hall et al. (2001) analyze a variety of problems with different machine environments where a set of available times at which batches may be delivered is fixed before the schedule is determined. There is apparently no literature that addresses the objective we consider, which is the total batch delivery cost plus a scheduling cost based on the delivery time of the batch. This objective is a natural extension of the classical scheduling literature to allow for batch delivery cost. Moreover, we demonstrate that it can be used effectively to model issues of coordinated decision making between different stages of a supply chain.

An important example of decision making that affects both a supplier and a manufacturer is the delivery process between them. The supplier processes jobs and delivers them to the manufacturer. The manufacturer may prefer to receive frequent deliveries of small batches from the supplier, because this will enable the manufacturer to achieve better resource utilization. However, the supplier may be reluctant to deliver very frequently because of the resulting high delivery cost. Furthermore, the manufacturer may prefer to receive parts earlier rather than later because this enlarges the manufacturer's scheduling options and improves utilization. However, because of production constraints at the supplier, its scheduling decisions may prioritize the processing of certain parts, perhaps destined for different customers, over others. Importantly, the scheduling decisions must be coordinated with the related batching and delivery decisions. We first investigate this decision problem from the viewpoint of the supplier.

We similarly consider the manufacturer's problem in a supply chain. The manufacturer also has downstream customers. The decision problem faced by the manufacturer is therefore very similar to that faced by the supplier. Indeed, the only difference is that the scheduling, batching, and delivery decisions made by the supplier define *batch release dates*, before which the manufacturer cannot begin work on any job in that batch. We demonstrate that these batch release dates can be incorporated into scheduling, batching, and delivery models.

The batch release dates defined for the manufacturer by the supplier's decisions may be less than ideal from the viewpoint of the manufacturer. This suggests that it may be mutually advantageous for the supplier and manufacturer to cooperate in developing combined models which incorporate the total costs of both parties. Therefore, we also describe and analyze models for cooperative decision making between the supplier and the manufacturer.

This paper is organized as follows. In §2, we describe our notation and classification scheme. We also provide some general results for all the scheduling objectives considered here, and we present an overview of our algorithmic and computational complexity results. Section 3 considers the minimization of the total scheduling and delivery cost, using a variety of classical scheduling objectives, from the viewpoint of the supplier. In §4, we consider the same problems from the viewpoint of the manufacturer. The problem of minimizing the total system cost of a supplier and a manufacturer who cooperate is studied in §5. The potential benefits from such cooperation, and practical mechanisms for achieving it, are discussed in §6. Section 7 contains a conclusion and some suggestions for future work.

2. PRELIMINARIES

In this section, we describe our notation and assumptions, and provide some general results that simplify the subsequent analysis. We then present an overview of the results in the paper.

2.1. Notation and Classification

We consider three categories of problems that arise in an arboresescent supply chain. In the *supplier's problem*, n^S jobs are to be scheduled on a single machine, M_S, by the supplier $\mathcal{S}$. Each job is produced for one of G manufacturers $\mathcal{M}_1, \ldots, \mathcal{M}_G$, and the jobs for each manufacturer are delivered in batches. The supplier's problem is illustrated in Figure 1, where "$*$" indicates that scheduling, batching, and delivery decisions by the supplier are required. Similarly, in the *manufacturer's problem*, one of the manufacturers (without loss of generality, we assume that this is $\mathcal{M}_1$) uses jobs from the supplier to produce n^M jobs on a single machine, M_M, for customers $\mathcal{C}_1, \ldots, \mathcal{C}_H$. These jobs are scheduled and delivered in batches to the customers, as illustrated in Figure 2. Finally, in the *combined problem*, we are required to find an overall schedule for the supplier $\mathcal{S}$ and manufacturer $\mathcal{M}_1$ that creates batches to be delivered from the supplier to each of the manufacturers, and from

Figure 1. Structure of the supplier's problems.

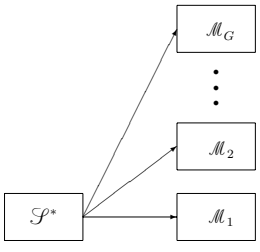
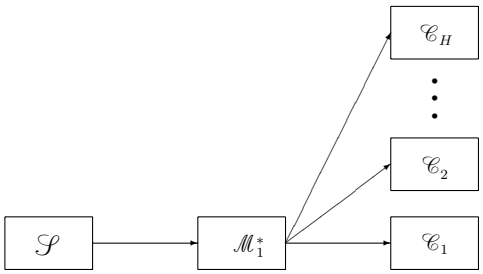


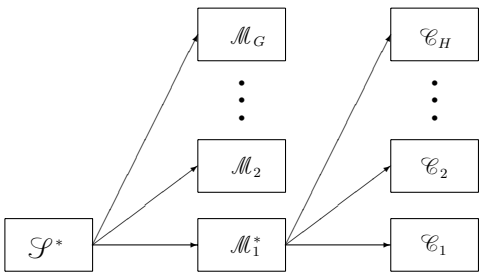
Figure 2. Structure of the manufacturer's problems.



manufacturer $\mathscr{M}_1$ to the customers. Figure 3 illustrates the structure of the combined problem.

We describe the notation for the *parameters* in our models, starting with the supplier's problem. Let $N^S = \{1, \ldots, n^S\}$ denote the set of jobs. We refer to the jobs processed for manufacturer $\mathscr{M}_g$, as $(1, g), \ldots, (n_g, g)$, for $g = 1, \ldots, G$. Job (j, g) has a processing time p^S_{jg} on machine M_s, for $j = 1, \ldots, n_g$. In some of the problems, job (j, g) has a weight (or value) w^S_{jg} and a due date d^S_{jg}. For the manufacturer's problem, the set of jobs is $N^M = \{1, \ldots, n^M\}$, and the jobs for customer $\mathscr{C}_h$ are $(1, h), \ldots, (n_h, h)$, for $h = 1, \ldots, H$. Job (j, h) has a processing time p^M_{jh} and, where relevant, has a weight w^M_{jh} and a due date d^M_{jh}, for $j = 1, \ldots, n_h$. The time when job (j, h) is delivered from the supplier to the manufacturer defines a release date r^M_{jh}, which is the earliest time when the manufacturer can start to process this job. For the combined problem, the set of jobs is $N^C = \{1, \ldots, n^C\}$. The jobs processed for customer $\mathscr{C}_h$ are $(1, h), \ldots, (n_h, h)$, for $h = 1, \ldots, H$, and these two-stage jobs require first machine M_S and then machine M_M. Further, the jobs processed for manufacturer $\mathscr{M}_g$ are $(1, H+g-1), \ldots, (n_{H+g-1}, H+g-1)$, for $g = 2, \ldots, G$, and these single-stage jobs require machine M_S only. Job (j, h) has a processing time p^S_{jh} on machine M_S and a processing time p^M_{jh} on machine M_M, and where relevant has a weight w^M_{jh} and a due date d^M_{jh}, for $h = 1, \ldots, H$ and $j = 1, \ldots, n_h$. Also, job (j, g) has a processing time p^S_{jg} on machine M_S, and where relevant has a weight w^S_{jg} and a due date d^S_{jg}, for $g = H+1, \ldots, H+G-1$ and $j = 1, \ldots, n_g$. We assume throughout that all processing times, weights, and release dates are positive integers, and that G and H are fixed so that they do not form part of the input data for a particular instance. Also, let P denote the sum of all the processing times in the specified problem, $W^S = \sum_{g=1}^{G} \sum_{j=1}^{n_g} w^S_{jg}$ and $W^M = \sum_{h=1}^{H} \sum_{j=1}^{n_h} w^M_{jh}$. Where there is no ambiguity, we write n instead of n^S or n^M or n^C, and W instead of W^S or W^M.

Figure 3. Structure of the combined problems.



We assume that the due dates are set as follows. Customers specify due dates by which the corresponding jobs should ideally be delivered from the manufacturer. To specify a due date for the supplier, the manufacturer assumes a specific (but not necessarily optimal) schedule for the processing of its jobs. This is typically a schedule in which all jobs can be delivered to the customers on time. Then, the due date of each job at the supplier is equal to the start time of that job in the manufacturer's given schedule.

A group of jobs forms a *batch* for the supplier if all of these jobs are delivered from the supplier to a single manufacturer at the same time. A batch for the manufacturer is defined analogously. Let D^S_g and D^M_h denote the nonnegative cost of delivering each batch from the supplier to manufacturer $\mathscr{M}_g$ for $g = 1, \ldots, G$, and from manufacturer $\mathscr{M}_1$ to customer $\mathscr{C}_h$ for $h = 1, \ldots, H$, respectively, independent of the contents of the batch.

Let the *variable* σ denote a supplier's schedule. Then we define:

$C^S_j(\sigma)$ = the time at which job j is delivered to the relevant manufacturer;
$F^S_j(\sigma) = C^S_j(\sigma)$, the flow time of job j;
$L^S_j(\sigma) = C^S_j(\sigma) - d^S_j$, the lateness of job j;

$$U^S_j(\sigma) = \begin{cases} 0 & \text{if job } j \text{ is delivered to the relevant manufacturer by its due date,} \\ 1 & \text{if job } j \text{ is late;} \end{cases}$$

$y^S_g(\sigma)$ = the number of deliveries to manufacturer $\mathscr{M}_g$ in σ.

The quantities $C^M_j(\sigma)$, $L^M_j(\sigma)$, $U^M_j(\sigma)$, and $y^M_h(\sigma)$ are defined analogously for a manufacturer's schedule σ, and $F^M_j(\sigma) = C^M_j(\sigma) - r^M_j$. Further, for the combined problem, we define $C^C_j(\sigma)$ to be $C^S_j(\sigma)$ for the single-stage jobs j that are produced for manufacturers $\mathscr{M}_2, \ldots, \mathscr{M}_G$, and $C^M_j(\sigma)$ for the two-stage jobs that are produced for customers $\mathscr{C}_1, \ldots, \mathscr{C}_H$, from which the analogous variables $F^C_j(\sigma) = C^C_j(\sigma)$, $L^C_j(\sigma)$, and $U^C_j(\sigma)$ are computed. When there is no ambiguity, we simplify $C^S_j(\sigma)$, $F^S_j(\sigma)$, $L^S_j(\sigma)$, $U^S_j(\sigma)$, and $y^S_g(\sigma)$ to C^S_j, F^S_j, L^S_j, U^S_j, and y^S_g, respectively, and make corresponding simplifications for the manufacturer's problem and the combined problem. We also refer to C^S_j and C^M_j as the completion times of job j (as perceived by the manufacturer and customer, respectively). Note that for any two-stage job j, $F^C_j = F^S_j + F^M_j$ when $r^M_j = C^S_j = F^S_j$, as assumed in our analysis.

The scheduling objectives which we consider are classical ones motivated either by internal costs or by external costs or target dates. In the absence of due dates, we consider total (weighted or unweighted) completion or flow-time objectives which model internal holding costs. Because we are considering a supply chain which ultimately delivers jobs to a customer, it is natural to use the definitions of C^S_j, C^M_j, and C^C_j in the previous paragraph. Where due dates are specified, we consider two alternative

assumptions about late jobs. First, in models which minimize the maximum lateness, $L_{\max}$, we assume that late jobs are produced and delivered. The objective function is used to ensure that the due dates are achieved as closely as possible. Second, in models which minimize the (weighted or unweighted) number of late jobs, we assume that a job which would be late is neither produced nor delivered. This assumption is relevant where late deliveries are not accepted. Without this assumption, the fact that a batch may contain both on-time and late jobs may significantly complicate solution procedures for the models we consider. Both the maximum lateness and the number of late jobs are standard objectives in the scheduling literature.

The standard classification scheme for scheduling problems (Graham et al. 1979) is $\alpha|\beta|\gamma$, where α indicates the scheduling environment, β describes the job characteristics or restrictive requirements, and γ defines the objective function to be minimized. Under α, in every case we have "1", which denotes a single machine. Under β, we may have "r_j", which denotes that each of the manufacturer's jobs has a release date r_{jh}^M that defines the earliest time at which job (j, h) becomes available for processing. The objective functions that we consider under γ comprise a delivery cost and a scheduling cost. The delivery cost for the supplier is $\sum D_g^S y_g^S$ and for the manufacturer is $\sum D_h^M y_h^M$. The scheduling cost is

$\sum(w_j)F_j^X$ = the total (weighted) flow time of the jobs;
$L_{\max}^X = \max_{j\in N}\{C_j^X - d_j^X\}$, the maximum lateness of the jobs;
$\sum(w_j)U_j^X$ = the total (weighted) number of late jobs;

where $X = S$ for the supplier, $X = M$ for the manufacturer, and $X = C$ for the combined problem. For the combined problem, we consider the total delivery cost $\sum D_g^S y_g^S + \sum D_h^M y_h^M$, the supplier's scheduling cost for those jobs of N^S that are produced for manufacturers $\mathcal{M}_2, \ldots, \mathcal{M}_G$, and the scheduling cost incurred by manufacturer $\mathcal{M}_1$ for all jobs of N^M. Because the combined problem considers the complete system as illustrated in Figure 3, the supplier's scheduling cost for the jobs of N^S that are produced for manufacturer $\mathcal{M}_1$ is assumed to represent a performance measure for work in process, and hence does not affect the total system cost. Note that minimizing $\sum(w_j)C_j^X$ is equivalent to minimizing $\sum(w_j)F_j^X$, so we do not consider the total (weighted) completion time objective explicitly.

2.2. General Assumptions and Properties

For problems in which the scheduling objective is to minimize the number of late jobs, we assume that it is sufficient to construct a delivery schedule for the on-time jobs only, and the delivery cost of the late jobs is ignored.

We present some results that apply to various problems, irrespective of the scheduling objective. The first result eliminates inserted idle time from the schedules we construct, and requires no proof.

LEMMA 1. *There exists an optimal schedule such that*:

(a) *In any of the supplier's and combined problems which we consider, there is no idle time between the jobs on machine* M_S.

(b) *In any of the manufacturer's problems which we consider, each job starts processing on machine* M_M *either at its release date, or immediately after another job.*

The next result restricts the choice of times at which batch deliveries are scheduled. A proof is provided by Hall and Potts (2000).

LEMMA 2. *There exists an optimal schedule with the following properties*:

(a) *In any problem that we consider, each delivery from the supplier* $\mathcal{S}$ *to any manufacturer* (*respectively, each delivery from manufacturer* $\mathcal{M}_1$ *to any customer*) *occurs when some job destined for the relevant manufacturer* (*respectively, customer*) *completes processing.*

(b) *In any of the supplier's or combined problems that we consider, a delivery from the supplier* $\mathcal{S}$ *to some manufacturer* $\mathcal{M}_g$ *that occurs when some job* (j, g) *completes processing comprises a group of jobs destined for manufacturer* $\mathcal{M}_g$, *including* (j, g), *that are processed consecutively on machine* M_S.

The third result restricts the processing orders on machines M_S and M_M in combined problems. Again, we refer to Hall and Potts (2000) for a proof.

LEMMA 3. *In any of the combined problems that we consider, there exists an optimal schedule in which the jobs on machine* M_M *are processed in the same order as the corresponding jobs are processed on machine* M_S.

Schedules that satisfy Lemma 3 are called *permutation schedules*.

The final result in this section provides a complexity hierarchy between a classical scheduling problem $\alpha|\beta|\gamma$, a supplier's problem $\alpha|\beta|\gamma^S$, a manufacturer's problem $\alpha|\beta, r_j|\gamma^M$, and a combined problem $\alpha|\beta|\gamma^C$. Given two problems P_1 and P_2, let the notation $P_1 \propto P_2$ denote that P_2 is a generalization of P_1.

THEOREM 1. $\alpha|\beta|\gamma \propto \alpha|\beta|\gamma^S \propto \alpha|\beta, r_j|\gamma^M$ *and* $\alpha|\beta|\gamma^S \propto \alpha|\beta|\gamma^C$.

PROOF. If $D_g^S = 0$ for $g = 1, \ldots, G$, then without loss of generality all batches in problem $\alpha|\beta|\gamma^S$ contain a single job, and consequently problems $\alpha|\beta|\gamma$ and $\alpha|\beta|\gamma^S$ are equivalent. Also, when $r_j^M = 0$ for $j \in N^M$, problems $\alpha|\beta|\gamma^S$ and $\alpha|\beta, r_j|\gamma^M$ are equivalent. Finally, if $D_h^M = 0$ for $h = 1, \ldots, H$ and all the manufacturer's processing times are zero, then problems $\alpha|\beta|\gamma^S$ and $\alpha|\beta|\gamma^C$ are equivalent. □

2.3. Overview of the Results

Because the cost and the feasibility of a schedule can be evaluated in $O(n)$ time, the recognition versions of all the problems considered in this paper belong to the class *NP*. Table 1 presents a summary of our results for the supplier's,

manufacturer's, and combined problems. For the manufacturer's problems, our polynomial time algorithms are derived under the assumption of batch consistency, which means that if some job i is delivered to the manufacturer before another job j, then i cannot be scheduled for delivery by the manufacturer strictly later than j. For various manufacturer's problems within Table 1, we also assume SPT-batch consistency, EDD-batch consistency, or EDD-batch consistency for the on-time jobs, which means that jobs with the same release date that are destined for the same customer are sequenced in SPT order (nondecreasing order of processing times), in EDD order (nondecreasing order of due dates), or in EDD order for the on-time jobs, respectively. For the combined problems, we assume a given ordering of the jobs for each of the manufacturers $\mathcal{M}_2, \ldots, \mathcal{M}_G$ and for each of the customers $\mathcal{C}_1, \ldots, \mathcal{C}_H$. All these assumptions are motivated and discussed in the relevant section of the paper. The first column of Table 1 specifies the scheduling objective. The second, third, and fourth columns show the running time of the fastest known polynomial or pseudopolynomial time algorithm for the supplier's, manufacturer's, and combined problem, respectively, if such an algorithm exists. Otherwise, we use *UNPC* or *BNPC* to indicate that the equivalent recognition version of a problem is *NP*-complete with respect to a unary or binary encoding of the data, respectively. Related definitions can be found in Garey and Johnson (1979). Included in the appropriate cell is a reference to where a proof of that result can be found.

3. THE SUPPLIER'S PROBLEM

In this section, we describe models for minimizing the sum of scheduling and delivery cost in various environments, from the viewpoint of the supplier. For some of the models, we establish the processing order of jobs for each manufacturer in an optimal schedule, and then use dynamic programming to form batches for delivery. Our dynamic programs are forward algorithms that append either a single job or a batch to a previously constructed partial schedule of jobs.

3.1. Sum of Flow Times

LEMMA 4. *For problem* $1\|\sum F_j^S + \sum D_g^S y_g^S$, *the cost is minimized by sequencing the jobs for each manufacturer* $\mathcal{M}_g$ *according to a shortest processing time* (*SPT*) *rule.*

PROOF. The result is established by a standard job interchange argument. □

As a result of Lemma 4, we assume throughout this subsection that the jobs for each manufacturer $\mathcal{M}_g$ are indexed in SPT order, so that $p_{1g}^S \leqslant \cdots \leqslant p_{n_g,g}^S$ for $g = 1, \ldots, G$. Generalizing ideas from Albers and Brucker (1993) for a similar problem with $G = 1$, we propose the following dynamic programming algorithm to solve problem $1 \| \sum C_j^S + \sum D_g^S y_g^S$.

Algorithm SF

Value Function

$f(q) = f(q_1, \ldots, q_G) =$ the minimum total cost for processing and delivering jobs $(1, g), \ldots, (q_g, g)$ for $g = 1, \ldots, G$, with the last delivery at time $\sum_{g=1}^{G} \sum_{j=1}^{q_g} p_{jg}^S$, where $0 \leqslant q_g \leqslant n_g$.

Boundary Condition

$f(0, \ldots, 0) = 0.$

Optimal Solution Value

$f(n_1, \ldots, n_G).$

Recurrence Relation

$f(q) = \min_{(q_g', g) \in J} \{(q_g - q_g')T + D_g^S + f(q')\},$

Table 1. The complexity of supply chain scheduling problems.

Scheduling Objective	Supplier's Problem		Manufacturer's Problem		Combined Problem	
$\sum C_j$ or $\sum F_j$	$O(n^{G+1})$	Thm 2	*UNPC*	Thm 1	*UNPC*	Thm 1
			$O(n^{3H})^{\S}$	Thm 7	$O(n^{2G+7H-2})^{\sharp}$	Thm 11
$\sum w_j C_j$ or $\sum w_j F_j$	*UNPC*	Thm 3	*UNPC*	Thm 1	*UNPC*	Thm 1
$L_{\max}$	$O(n^{2G+1})$	Thm 4	*UNPC*	Thm 1	*UNPC*	Thm 1
			$O(n^{4H})^{\dagger}$	Thm 8	$O(n^{3G+8H-2})^{\P}$	Thm 12
$\sum U_j$	$O(n^{2G+2})$	Thm 5	*UNPC*	Thm 1	*UNPC*	Thm 1
					BNPC$^{\|}$	Thm 13
			$O(n^{3H+1})^{\ddagger}$	Thm 9	$O(n^{G+2H-1}P^3)^{\|}$	Thm 14
	BNPC	Thm 6	*UNPC*	Thm 1	*UNPC*	Thm 1
$\sum w_j U_j$			*BNPC*‡	Thm 10	*BNPC*$^{\|}$	Thm 13
	$O(n^{2G+1}W)$	Thm 6	$O(n^{3H}W)^{\ddagger}$	Thm 10	$O(n^{G+2H-1}P^3)^{\|}$	Thm 14

§Result applies for SPT-batch consistency.
♯Result applies for total SPT within groups sequencing.
†Result applies for EDD-batch consistency.
¶Result applies for EDD within groups sequencing.
‡Result applies for EDD-batch consistency for the on-time jobs.
‖Result applies for EDD within groups sequencing for the on-time jobs.

where

$$J = \{(q'_g, g) \mid 1 \leqslant g \leqslant G, q_g > 0, 0 \leqslant q'_g < q_g\},$$

$$T = \sum_{g=1}^{G} \sum_{j=1}^{q_g} p^S_{jg} \quad \text{and}$$

$$q' = (q_1, \ldots, q_{g-1}, q'_g, q_{g+1}, \ldots, q_G).$$

The recurrence relation selects a batch $\{(q'_g + 1, g), \ldots, (q_g, g)\}$ of jobs to be delivered to manufacturer $\mathcal{M}_g$ at time T. Each of these jobs contributes a flow time of T to the total scheduling cost, and a delivery cost of D^S_g is also incurred.

THEOREM 2. *Algorithm SF finds an optimal schedule for problem* $1 \parallel \sum C^S_j + \sum D^S_g y^S_g$ *in* $O(n^{G+1})$ *time.*

PROOF. The dynamic program exploits properties of an optimal schedule from part (a) of Lemma 1, Lemma 2, and Lemma 4. There are $O(n^G)$ states $(q_1, \ldots, q_G)$, and for each state the recurrence relation requires $O(n)$ time. Therefore, the overall time complexity of Algorithm SF is $O(n^{G+1})$. □

However, we now show that the weighted version of this problem is intractable, even when there is only one manufacturer.

THEOREM 3. *The recognition version of problem* $1 \parallel \sum w_j F^S_j + \sum D^S_g y^S_g$ *is unary NP-complete, even for* $G = 1$.

PROOF. We use a reduction from the following problem which is known to be unary *NP*-complete.

3-PARTITION (Garey and Johnson 1979). Given $3m$ elements with integer sizes $a_1, \ldots, a_{3m}$, where $\sum_{i=1}^{3m} a_i = mz$ and $z/4 < a_i < z/2$, for $i = 1, \ldots, 3m$, does there exist a partition $S_1, \ldots, S_m$ of the index set $\{1, \ldots, 3m\}$, such that $|S_j| = 3$ and $\sum_{i \in S_j} a_i = z$, for $j = 1, \ldots, m$? We assume without loss of generality that, if there exists a solution to 3-Partition, then the elements are numbered such that $a_{3i-2} + a_{3i-1} + a_{3i} = z$, for $i = 1, \ldots, m$.

Consider an instance of the recognition version of problem $1 \parallel \sum w_j F^S_j + \sum D^S_g y^S_g$ defined by
$n = 3m$,
$G = 1$,
$p^S_{i,1} = w^S_{i,1} = a_i$, for $i = 1, \ldots, n$,
$D^S_1 = z^2/2$,
$C = m(m+2)z^2/2$,
where C is a threshold cost. We prove that there exists a schedule for this instance of $1 \parallel \sum w_j F^S_j + \sum D^S_g y^S_g$ with cost less than or equal to C if and only if there exists a solution to 3-Partition.

($\Rightarrow$) Consider the sequence $(1, \ldots, n)$ with deliveries when job $3j$ completes processing, for $j = 1, \ldots, m$. This schedule yields $\sum w^S_j F^S_j + \sum D^S_g y^S_g = z \cdot z + z \cdot 2z + \cdots + z \cdot mz + mz^2/2 = C$.

($\Leftarrow$) We first show that a cost minimizing solution with k batches has equal total processing time in each batch. We can formulate this problem as

$$\begin{aligned} \text{minimize} \quad & x_1^2 + (x_1 + x_2)x_2 + \cdots + (x_1 + \cdots + x_k)x_k \\ & + kz^2/2 \\ \text{subject to} \quad & x_1 + \cdots + x_k = mz, \end{aligned}$$

where x_i is the total processing time in batch i, for $i = 1, \ldots, k$. The objective function can be expressed as

$$\left(\sum_{i=1}^{k} x_i\right)^2 \Big/ 2 + \sum_{i=1}^{k} x_i^2 \Big/ 2 + kz^2/2$$
$$= m^2 z^2/2 + \sum_{i=1}^{k} x_i^2 \Big/ 2 + kz^2/2.$$

This second term is minimized by setting $x_1 = \cdots = x_k = mz/k$, and any other solution yields a higher cost. Thus, the total cost for k deliveries is greater than or equal to

$$m^2 z^2/2 + m^2 z^2/(2k) + kz^2/2.$$

This expression achieves a minimum with respect to k only at $k = m$, and this minimum value is equal to C. This implies that the schedule has m batches, each with a total processing time of z. From the definition of the elements in 3-Partition, each batch with a total processing time of z must contain exactly three jobs. Thus, a solution to 3-Partition exists. □

3.2. Maximum Lateness

In this subsection, we adapt Algorithm SF of the previous subsection to the supplier's problem with a scheduling objective of minimizing the maximum lateness.

LEMMA 5. *For problem* $1 \parallel L^S_{\max} + \sum D^S_g y^S_g$, *the cost is minimized by sequencing the jobs for each manufacturer* $\mathcal{M}_g$ *according to an earliest due date* (*EDD*) *rule.*

PROOF. The result is established by a standard job insertion argument. □

As a result of Lemma 5, we assume throughout this subsection that the jobs for each manufacturer $\mathcal{M}_g$ are indexed in EDD order, so that $d^S_{1,g} \leqslant \cdots \leqslant d^S_{n_g,g}$ for $g = 1, \ldots, G$. For problem $1 \parallel L^S_{\max} + \sum D^S_g y^S_g$, our dynamic programming algorithm uses the maximum lateness of the partial schedule as a function value. Also, the delivery cost is evaluated from state variables $y_1, \ldots, y_G$, which specify the number of deliveries to the respective manufacturers. The minimum total cost can therefore be computed after applying the recursion. We note that a similar dynamic program for a special case with $G = 1$ is derived by Webster and Baker (1995).

Algorithm SL

Value Function

$f(q, y) = f(q_1, \ldots, q_G, y_1, \ldots, y_G)$ = the minimum value of the maximum lateness for processing and delivering jobs $(1, g), \ldots, (q_g, g)$, using y_g deliveries for manufacturer $\mathcal{M}_g$, for $g = 1, \ldots, G$, with the last delivery at time $\sum_{g=1}^{G} \sum_{j=1}^{q_g} p_{jg}^S$, where $0 \leqslant y_g \leqslant q_g \leqslant n_g$.

Boundary Condition

$f(0, \ldots, 0, 0 \ldots, 0) = -\infty$.

Optimal Solution Value

$$\min_{y_1, \ldots, y_g} \left\{ f(n_1, \ldots, n_G, y_1, \ldots, y_G) + \sum_{g=1}^{G} D_g^S y_g \right\},$$

where the minimization is over $1 \leqslant y_g \leqslant n_g$ for $g = 1, \ldots, G$.

Recurrence Relation

$f(q, y) = \min_{(q'_g, g) \in J} \{\max\{T - d^S_{q'_g+1, g}, f(q', y')\}\}$,
where $y' = (y_1, \ldots, y_{g-1}, y_g - 1, y_{g+1}, \ldots, y_G)$ and J, T, and q' are defined as in Algorithm SF.

The recurrence relation selects a batch $\{(q'_g + 1, g), \ldots, (q_g, g)\}$ of jobs to be delivered to manufacturer $\mathcal{M}_g$ at time T. From the EDD indexing of the jobs, job $(q'_g + 1, g)$ has the smallest due date and hence also the maximum lateness in this batch. The lateness of job $(q'_g + 1, g)$ is compared with the maximum lateness of the jobs that have been scheduled earlier.

THEOREM 4. *Algorithm SL finds an optimal schedule for problem* $1 \parallel L^S_{\max} + \sum D^S_g y^S_g$ *in* $O(n^{2G+1})$ *time.*

PROOF. The dynamic program exploits properties of an optimal schedule from part (a) of Lemma 1, Lemma 2, and Lemma 5. There are $O(n^{2G})$ states $(q_1, \ldots, q_G, y_1, \ldots, y_G)$, and for each state the recurrence relation requires $O(n)$ time. Therefore, the overall time complexity of Algorithm SL is $O(n^{2G+1})$. □

3.3. Number of Late Jobs

First, we state without proof the following adaptation of Lemma 5 to weighted number of late jobs problems.

LEMMA 6. *For problem* $1 \parallel \sum w_j U_j^S + \sum D_g^S y_g^S$, *the cost is minimized by sequencing the on-time jobs for each manufacturer according to an earliest due date (EDD) rule.*

As a result of Lemma 6, we assume throughout this subsection that the jobs for each manufacturer $\mathcal{M}_g$ are indexed in EDD order. For problem $1 \parallel \sum U_j^S + \sum D_g^S y_g^S$, we describe a dynamic programming algorithm that either appends a job to some previous schedule of on-time jobs, or specifies that this job is late. A subsequent decision is made about whether a batch delivery is scheduled on completion of an appended on-time job. To allow us to check that all jobs of the current batch are on-time when the batch is delivered, we store the index $(j, \bar{g})$ of the lowest-indexed job in this batch as a state variable. We note that because the jobs are indexed in EDD order, all the jobs in the batch are on time if and only if the lowest-indexed job is. The makespan of the current partial schedule is stored as the function value, while the number of late jobs u is treated as a state variable.

Algorithm SU

Value Function

$f(q, y, u, j, \bar{g}) = f(q_1, \ldots, q_G, y_1, \ldots, y_G, u, j, \bar{g})$ = the minimum makespan for processing the on-time jobs from $(1, g), \ldots, (q_g, g)$ for $g = 1, \ldots, G$, given that for manufacturer $\mathcal{M}_g$ all on-time jobs are delivered using y_g deliveries for $g = 1, \ldots, \bar{g} - 1, \bar{g} + 1, \ldots, G$, for manufacturer $\mathcal{M}_{\bar{g}}$ all processed jobs from $(1, \bar{g}), \ldots, (j - 1, \bar{g})$ are delivered using $y_{\bar{g}} - 1$ deliveries if $y_{\bar{g}} > 0$, $(j, \bar{g})$ is the first (earliest due date) job in the last batch of the current partial schedule that is not yet scheduled for delivery if $j > 0$, and the total number of late jobs is u, where $0 \leqslant y_g \leqslant q_g \leqslant n_g$, $0 \leqslant u \leqslant n$, $0 \leqslant j \leqslant q_{\bar{g}}$ and $1 \leqslant \bar{g} \leqslant G$. If $j > 0$ and $f(q, y, u, j, \bar{g}) > d^S_{j\bar{g}}$, then we define $f(q, y, u, j, \bar{g}) = \infty$.

Boundary Condition

$f(0, \ldots, 0, 0, \ldots, 0, 0, 0, 0) = 0$.

Optimal Solution Value

$$\min \left\{ u + \sum_{g=1}^{G} D_g^S y_g \,\middle|\, \min_{(j,\bar{g}) \in N^S} \{f(n_1, \ldots, n_G, y_1, \ldots, y_G, u, j, \bar{g})\} < \infty, 0 \leqslant u \leqslant n, 0 \leqslant y_g \leqslant n_g \text{ for } g = 1, \ldots, G \right\}.$$

Recurrence Relation

$$f(q, y, u, j, \bar{g}) = \min \begin{cases} f(q', y, u - 1, j, \bar{g}) \\ p^S_{q_{\bar{g}}, \bar{g}} + f(q', y, u, j, \bar{g}), & \text{if } 0 < j < q_{\bar{g}} \text{ and } f(q', y, u, j, \bar{g}) + p^S_{q_{\bar{g}}, \bar{g}} \leqslant d^S_{j\bar{g}}, \\ \min_{(j', \bar{g}') \in J} \{p^S_{q_{\bar{g}}, \bar{g}} + f(q', y', u, j', \bar{g}')\}, & \text{if } j = q_{\bar{g}}, \end{cases}$$

where

$q' = (q_1, \ldots, q_{\bar{g}-1}, q_{\bar{g}} - 1, q_{\bar{g}+1}, \ldots, q_G)$,
$y' = (y_1, \ldots, y_{\bar{g}-1}, y_{\bar{g}} - 1, y_{\bar{g}+1}, \ldots, y_G)$, and
$J = \{(j', \bar{g}') \mid 0 \leqslant \bar{g}' \leqslant G, j' = 0 \text{ if } \bar{g}' = 0, 1 \leqslant j' \leqslant q_{\bar{g}} - 1 \text{ if } \bar{g}' = \bar{g} \neq 0, 1 \leqslant j' \leqslant q_{\bar{g}'} \text{ if } \bar{g}' \neq \bar{g} \text{ and } \bar{g}' \neq 0, f(q', y', u, j', \bar{g}') + p^S_{q_{\bar{g}}, \bar{g}} \leqslant d^S_{q_{\bar{g}}, \bar{g}}\}$.

The first term of the minimization in the recurrence relation schedules job $(q_{\bar{g}}, \bar{g})$ to be late. The second term schedules job $(q_{\bar{g}}, \bar{g})$ to be on time and belonging to the current batch of jobs for customer $\mathcal{M}_{\bar{g}}$, provided that job $(q_{\bar{g}}, \bar{g})$ can be completed no later than time $d^S_{j\bar{g}}$ so that job $(j, \bar{g})$ can be dispatched by its due date. No decision has yet been made about when to deliver the batch containing jobs $(j, \bar{g})$ and $(q_{\bar{g}}, \bar{g})$. The third term schedules a delivery for a batch containing $(j', \bar{g}')$ as its first job if $j' > 0$, starts a new batch with job (q_g, g) which is scheduled to be on time, and accounts

for a delivery that will eventually need to be made to manufacturer $\mathcal{M}_{\bar{g}}$. In the definition of J, we distinguish the case where customers $\bar{g}'$ and $\bar{g}$ are the same, and therefore the first job $(j', \bar{g}')$ in the delivered batch satisfies $j' \leqslant q_{\bar{g}} - 1$, from the case where those customers are different, and therefore the first job can also be $(q_{\bar{g}'}, \bar{g}')$.

THEOREM 5. *Algorithm SU finds an optimal schedule for problem* $1 \parallel \sum U_j^S + \sum D_g^S y_g^S$ *in* $O(n^{2G+2})$ *time.*

PROOF. The dynamic program exploits properties of an optimal schedule from part (a) of Lemma 1, Lemma 2, and Lemma 6. There are $O(n^{2G+2})$ states $(q_1, \ldots, q_G, y_1, \ldots, y_G, u, j, \bar{g})$. The first and second terms in the recurrence relation require constant time for each state. The third term requires $O(n)$ time for each of the $O(n^{2G+1})$ states for which $j = q_{\bar{g}}$. Therefore, the overall time complexity of Algorithm SU is $O(n^{2G+2})$. □

We now consider the corresponding problem with the weighted number of late jobs as a scheduling objective.

THEOREM 6. *An optimal solution for problem* $1 \parallel \sum w_j U_j^S + \sum D_g^S y_g^S$ *can be found in* $O(n^{2G+1}W)$ *time. The recognition version of this problem is binary* NP*-complete.*

PROOF. We can modify Algorithm SU by replacing the state variable u representing the number of late jobs by one that specifies the total weight of late jobs. The original time complexity of $O(n^{2G+2})$ in Algorithm SU becomes $O(n^{2G+1}W)$ with this modification. The second result follows from the binary *NP*-completeness of the recognition version of the classical problem $1 \parallel \sum w_j U_j$ (Karp 1972) and from Theorem 1. □

4. THE MANUFACTURER'S PROBLEM

Here we describe models for minimizing the sum of scheduling cost plus delivery cost from the viewpoint of one particular manufacturer, $\mathcal{M}_1$. In these models, the time at which each job j is delivered from the supplier to the manufacturer defines a release date r_j^M from the viewpoint of the manufacturer.

Unfortunately, neither the SPT ordering of Lemma 4 nor the EDD ordering of Lemmas 5 and 6 can be generalized to give a sequence of jobs for each customer here because the jobs have different release dates. However, by making some natural assumptions about the processing order of jobs for each customer, we are still able to provide optimal polynomial or pseudopolynomial time algorithms. The algorithms here are more complicated than those in §3. This is because part (b) of Lemma 2 does not generalize to batches that are delivered from a manufacturer to a customer, due to the presence of release dates.

A supplier's batch schedule and a manufacturer's batch schedule are *batch consistent* if for each pair of jobs (i, h) and (j, h) that are processed by the supplier $\mathcal{S}$ and manufacturer $\mathcal{M}_1$, where $1 \leqslant h \leqslant H$, $1 \leqslant i, j \leqslant n_h$, and $i \neq j$, whenever job (i, h) is in a strictly earlier batch than job (j, h) in the supplier's batch schedule, then job (i, h) is in an earlier batch or in the same batch as job (j, h) in the manufacturer's batch schedule. The algorithms developed by Ahmadi et al. (1992) for serial production processes with both discrete and batch processors provide schedules that meet this assumption. Moreover, this assumption is a natural simplification of the production planning process because it permits some scheduling decisions to be made without waiting for the next incoming batch. Another reason for making this assumption is that it reduces the need for resequencing and for related storage buffers.

4.1. Sum of Flow Times

From the unary *NP*-completeness of the recognition version of the classical problem $1|r_j|\sum F_j$ (Lenstra et al. 1977) and Theorem 1, the recognition version of problem $1|r_j|\sum F_j^M + \sum D_h^M y_h^M$ is also unary *NP*-complete.

We derive a dynamic programming algorithm for problem $1|r_j|\sum F_j^M + \sum D_h^M y_h^M$ under the assumption of batch consistency. We make the further assumption of *SPT-batch consistency*, in which jobs with the same release date (that is, jobs delivered in one batch by the supplier) and for the same customer are processed by the manufacturer in SPT order. Thus, for customer $\mathcal{C}_h$, we index the jobs such that $r_{1h}^M \leqslant \cdots \leqslant r_{n_h,h}^M$, for $h = 1, \ldots, H$, where $r_{jh}^M = r_{j+1,h}^M$ implies that $p_{jh}^M \leqslant p_{j+1,h}^M$ for $j = 1, \ldots, n_h - 1$. This assumption is motivated by a local optimization rule that minimizes the $\sum F_j^M$ objective within a batch.

We now provide a dynamic programming algorithm for problem $1|r_j|\sum F_j^M + \sum D_h^M y_h^M$ under the assumption of SPT-batch consistency. We need additional state variables to those in Algorithm SF, first because we consider the jobs one at a time here and therefore some jobs may have been processed and not yet delivered, and second because the presence of release dates may cause machine idle time which affects the makespan of the current partial schedule. We partition the partial schedule into *blocks*, where a block is a maximal set of jobs that are processed with no idle time in between. Thus, the first job in a block is either the first job in the schedule or is preceded by idle time, and starts processing at its release date.

Algorithm MF

Value Function

$f(q, s, b, \bar{h}) = f(q_1, \ldots, q_H, s_1, \ldots, s_H, b_1, \ldots, b_H, \bar{h}) =$ the minimum total cost for processing and delivering jobs $(1, h), \ldots, (s_h, h)$, for $h = 1, \ldots, H$ (where flow time is evaluated on the basis of zero release dates), with jobs $(s_h + 1, h), \ldots, (q_h, h)$ also processed for $h = 1, \ldots, H$, and with the last block containing jobs $(b_h + 1, h), \ldots, (q_{\bar{h}}, h)$ for $h = 1, \ldots, H$ and starting with job $(b_{\bar{h}} + 1, \bar{h})$ if $\bar{h} > 0$, where $0 \leqslant s_h \leqslant q_h \leqslant n_h$, $0 \leqslant b_h \leqslant q_h$, $b_{\bar{h}} < q_{\bar{h}}$ if $\bar{h} > 0$, and $0 \leqslant \bar{h} \leqslant H$.

Boundary Condition

$f(0, \ldots, 0, 0, \ldots, 0, 0, \ldots, 0, 0) = 0.$

Optimal Solution Value

$$\min_{(b_1,\ldots,b_H,\bar{h})\in B}\{f(n_1,\ldots,n_H,n_1,\ldots,n_H,b_1,\ldots,b_H,\bar{h})\} - \sum_{h=1}^{H}\sum_{j=1}^{n_h} r_{jh}^{M},$$

where $B=\{(b_1,\ldots,b_H,\bar{h}) \mid 0\leqslant b_h\leqslant n_h \text{ for } 1\leqslant h\leqslant H,\ b_{\bar{h}}<n_{\bar{h}},\ 1\leqslant\bar{h}\leqslant H\}$.

Recurrence Relation

$$f(q,s,b,\bar{h}) = \min\begin{cases} \min\limits_{h\in H_1}\{f(q',s,b,\bar{h})\} \\ \min\limits_{h\in H_2}\Big\{\min\limits_{0\leqslant s'_h<s_h}\{(q_h-s'_h)T+D_h^M+f(q',s',b,\bar{h})\}\Big\} \\ \min\limits_{(b',\bar{h}')\in B_1}\{f(q'',s,b',\bar{h}')\}, \quad \text{if } b=q'', s_{\bar{h}}<q_{\bar{h}} \\ \min\limits_{(b',\bar{h}')\in B_1}\Big\{\min\limits_{0\leqslant s''_{\bar{h}}<s_{\bar{h}}}\{(q_{\bar{h}}-s''_{\bar{h}})T+D_{\bar{h}}^M \\ \qquad +f(q'',s'',b',\bar{h}')\}\Big\}, \quad \text{if } b=q'', s_{\bar{h}}=q_{\bar{h}} \end{cases}$$

where

$$\begin{aligned} q' &= (q_1,\ldots,q_{h-1},q_h-1,q_{h+1}\ldots,q_H),\\ s' &= (s_1,\ldots,s_{h-1},s'_h,s_{h+1}\ldots,s_H),\\ q'' &= (q_1,\ldots,q_{\bar{h}-1},q_{\bar{h}}-1,q_{\bar{h}+1}\ldots,q_H),\\ s'' &= (s_1,\ldots,s_{\bar{h}-1},s''_{\bar{h}},s_{\bar{h}+1}\ldots,s_H),\\ H_1 &= \{h \mid 1\leqslant h\leqslant H,\ s_h<q_h,\ b_h<q_h,\ b_{\bar{h}}+1<q_h, \text{ if } h=\bar{h},\ r_{q_h,h}^M\leqslant T'\},\\ H_2 &= \{h \mid 1\leqslant h\leqslant H,\ s_h=q_h,\ b_h<q_h,\ b_{\bar{h}}+1<q_h, \text{ if } h=\bar{h},\ r_{q_h,h}^M\leqslant T'\},\\ B_1 &= \{(b'_1,\ldots,b'_H,\bar{h}') \mid 0\leqslant\bar{h}'\leqslant H,\ 0\leqslant b'_h\leqslant b_h \text{ for } h=1,\ldots,H, b'_{\bar{h}'}<b_{\bar{h}'}, \text{ if } \bar{h}'>0,\ r_{q_{\bar{h}},\bar{h}}^M>T'', \text{ if } \bar{h}'>0\},\\ T &= r_{b_{\bar{h}}+1,\bar{h}}^M+\textstyle\sum_{h=1}^{H}\sum_{j=b_h+1}^{q_h}p_{jh}^M,\\ T' &= r_{b_{\bar{h}}+1,\bar{h}}^M+\textstyle\sum_{l=1}^{H}\sum_{j=b_l+1}^{q'_l}p_{jl}^M, \text{ and}\\ T'' &= r_{b'_{\bar{h}'}+1,\bar{h}'}^M+\textstyle\sum_{l=1}^{H}\sum_{j=b'_l+1}^{q''_l}p_{jl}^M. \end{aligned}$$

In the recurrence relation, the first term in the minimization schedules job (q_h,h) to be processed at the end of the current block, and does not schedule a delivery upon completion of this job. The set H_1 ensures that $s_h<q_h$ so that job (q_h,h) is not delivered, $b_h<q_h$ so that job (q_h,h) is in the last block, and that there is no idle time immediately before job (q_h,h) is processed. The second term is similar, except that a batch $\{(s'_h+1,h),\ldots,(q_h,h)\}$ is scheduled for delivery to customer $\mathscr{C}_h$ when job (q_h,h) completes processing. In this case, the set H_2 ensures that $q_h=s_h$, so that the current state is consistent with such a delivery. The third and fourth terms in the recurrence relation correspond to the case where job $(q_{\bar{h}},\bar{h})$ is the last job in the current schedule and starts a new block. Specifically, the third term considers the case where $s_{\bar{h}}<q_{\bar{h}}$ so that no delivery is scheduled on completion of job $(q_{\bar{h}},\bar{h})$, whereas $s_{\bar{h}}=q_{\bar{h}}$ in the fourth term so that a batch delivery for jobs $\{(s''_{\bar{h}}+1,\bar{h}),\ldots,(q_{\bar{h}},\bar{h})\}$ is scheduled at time $T=r_{q_{\bar{h}},\bar{h}}+p_{q_{\bar{h}},\bar{h}}$. The set B_1 defines b' and $\bar{h}'$ for a previous state that allows the processing of job $(q_{\bar{h}},\bar{h})$ to start at its release date and be preceded by machine idle time.

THEOREM 7. *Algorithm MF finds an optimal SPT-batch consistent schedule for problem* $1|r_j|\sum F_j^M+\sum D_h^M y_h^M$ *in* $O(n^{3H})$ *time.*

PROOF. The dynamic program exploits properties of an optimal schedule from part (b) of Lemma 1, part (a) of Lemma 2, and the assumption of SPT-batch consistency. There are $O(n^{3H})$ states $(q,s,b,\bar{h})$. In the recurrence relation, the first term in the minimization requires constant time. The second term requires $O(n)$ time, but is only applied for the $O(n^{3H-1})$ states for which $s_h=q_h$. The third term requires $O(n^H)$ time, but is only applied for the $O(n^{2H})$ states for which $b_l=q_l$ for $l\neq\bar{h}$ and $b_{\bar{h}}=q_{\bar{h}}-1$. Similarly, the fourth term requires $O(n^{H+1})$ time, but is only applied for the $O(n^{2H-1})$ states for which $b_l=q_l$ for $l\neq\bar{h}$, $b_{\bar{h}}=q_{\bar{h}}-1$, and $s_{\bar{h}}=q_{\bar{h}}$. Therefore, the overall time complexity of Algorithm MF is $O(n^{3H})$. □

4.2. Maximum Lateness

From the unary *NP*-completeness of the recognition version of the classical problem $1|r_j|L_{\max}$ (Lenstra et al. 1977) and Theorem 1, the recognition version of problem $1|r_j|L_{\max}^M+\sum D_h^M y_h^M$ is also unary *NP*-complete. However, we provide a polynomial time algorithm for a restricted class of schedules.

We define *EDD-batch consistency* analogously to SPT-batch consistency in the previous subsection. Jobs with the same release date that are destined for the same customer are processed by the manufacturer in EDD order. Thus, for customer $\mathscr{C}_h$, we index the jobs such that $r_{1h}^M\leqslant\cdots\leqslant r_{n_h,h}^M$, for $h=1,\ldots,H$, where $r_{jh}^M=r_{j+1,h}^M$ implies that $d_{jh}^M\leqslant d_{j+1,h}^M$ for $j=1,\ldots,n_h-1$. This assumption is motivated by a local optimization rule that minimizes the $L_{\max}^M$ objective within a batch.

We now provide a dynamic programming algorithm for problem $1|r_j|L_{\max}^M+\sum D_h^M y_h^M$ under the assumption of EDD-batch consistency. Our algorithm is similar to Algorithm MF, although we need additional state variables to define the number of deliveries to each customer.

Algorithm ML

Value Function

$f(q,s,b,y,\bar{h}) = f(q_1,\ldots,q_H,s_1,\ldots,s_H,b_1,\ldots,b_H,y_1,\ldots,y_H,\bar{h})$ = the minimum value of the maximum lateness for processing and delivering jobs $(1,h),\ldots,(s_h,h)$ using y_h deliveries, for $h=1,\ldots,H$, with jobs $(s_h+1,h),\ldots,(q_h,h)$ also processed for $h=1,\ldots,H$, and with the last block containing jobs $(b_h+1,h),\ldots,(q_h,h)$ for $h=1,\ldots,H$ and starting with job $(b_{\bar{h}}+1,\bar{h})$ if $\bar{h}>0$, where $0\leqslant y_h\leqslant s_h\leqslant q_h\leqslant n_h, 0\leqslant b_h\leqslant q_h, b_{\bar{h}}<q_{\bar{h}}$ if $\bar{h}>0$, and $0\leqslant\bar{h}\leqslant H$.

Boundary Condition

$$f(0,\dots,0,0,\dots,0,0,\dots,0,0,\dots,0,0)=-\infty.$$

Optimal Solution Value

$$\min_{(b_1,\dots,b_H,y_1,\dots,y_H,\bar{h})\in B}\Big\{f(n_1,\dots,n_H,n_1,\dots,n_H,b_1,\dots,b_H,y_1,\dots,y_H,\bar{h})+\sum_{h=1}^{H}D_h^M y_h\Big\},$$

where

$$B=\big\{(b_1,\dots,b_H,y_1,\dots,y_H,\bar{h})\mid 0\leqslant b_h\leqslant n_h,\ 0\leqslant y_h\leqslant n_h \text{ for } 1\leqslant h\leqslant H, b_{\bar{h}}<n_{\bar{h}}, 1\leqslant\bar{h}\leqslant H\big\}.$$

Recurrence Relation

$$f(q,s,b,y,\bar{h})=\min\begin{cases}\min\limits_{h\in H_1}\{f(q',s,b,y,\bar{h})\}\\ \min\limits_{h\in H_2}\Big\{\min\limits_{0\leqslant s'_h<s_h}\Big\{\max\Big\{\max\limits_{s'_h+1\leqslant j\leqslant q_h}\{T-d_{jh}^M\},\ f(q',s',b,y',\bar{h})\Big\}\Big\}\Big\}\\ \min\limits_{(b',\bar{h}')\in B_1}\{f(q'',s,b',y,\bar{h}')\},\quad \text{if } b=q'', s_{\bar{h}}<q_{\bar{h}}\\ \min\limits_{(b',\bar{h}')\in B_1}\Big\{\min\limits_{0\leqslant s''_{\bar{h}}<s_{\bar{h}}}\Big\{\max\Big\{\max\limits_{s''_{\bar{h}}+1\leqslant j\leqslant q_{\bar{h}}}\{T-d_{j\bar{h}}^M\},\ f(q'',s'',b',y'',\bar{h}')\Big\}\Big\}\Big\},\quad \text{if } b=q'', s_{\bar{h}}=q_{\bar{h}}\end{cases}$$

where

$$y'=(y_1,\dots,y_{h-1},y_h-1,y_{h+1},\dots,y_H),$$
$$y''=(y_1,\dots,y_{\bar{h}-1},y_{\bar{h}}-1,y_{\bar{h}+1},\dots,y_H),$$

and q', s', q'', s'', H_1, H_2, B_1, T, T' and T'' are defined as in Algorithm MF.

The interpretation of the recurrence relation is similar to that of Algorithm MF.

THEOREM 8. *Algorithm ML finds an optimal EDD-batch consistent schedule for problem* $1|r_j|L_{\max}^M+\sum D_h^M y_h^M$ *in* $O(n^{4H})$ *time.*

PROOF. The dynamic program exploits properties of an optimal schedule from part (b) of Lemma 1, part (a) of Lemma 2, and the assumption of EDD-batch consistency. Because there are $O(n^{4H})$ states $(q,s,b,y,\bar{h})$, we can apply an analysis similar to that in the proof of Theorem 7 to show that the time complexity of Algorithm ML is $O(n^{4H})$. □

4.3. Number of Late Jobs

From the unary *NP*-completeness of the recognition version of the classical problem $1|r_j|\sum U_j$ (Lenstra et al. 1977) and Theorem 1, the recognition version of problem $1|r_j|\sum U_j^M+\sum D_h^M y_h^M$ is also unary *NP*-complete. Thus, we again restrict the class of schedules that we consider.

We assume *EDD-batch consistency for the on-time jobs*. Accordingly, we index the jobs such that $r_{1h}^M\leqslant\cdots\leqslant r_{n_h,h}^M$ for $h=1,\dots,H$, where $r_{jh}^M=r_{j+1,h}^M$ implies that $d_{jh}^M\leqslant d_{j+1,h}^M$ for $j=1,\dots,n_h-1$. This assumption is motivated by a local rule that achieves due dates for on-time jobs within a batch.

We now provide a dynamic programming algorithm for problem $1|r_j|\sum U_j^M+\sum D_h^M y_h^M$, under the assumption of EDD-batch consistency for the on-time jobs. Following the approach in Algorithm SU, we define the function value to be the makespan of the current partial schedule of on-time jobs. We also introduce state variables to store the undelivered on-time job with the smallest due date for each customer, to check that all jobs of a batch are on time when a delivery is scheduled.

Algorithm MU

Value Function

$f(q,v,y,u)=f(q_1,\dots,q_H,v_1,\dots,v_H,y_1,\dots,y_H,u)=$ the minimum makespan for processing the on-time jobs from $(1,h),\dots,(q_h,h)$ for $h=1,\dots,H$, given that y_h deliveries are used for each completed batch for customer $\mathscr{C}_h$ and (v_h,h) is the job with the smallest due date among the undelivered on-time jobs for customer $\mathscr{C}_h$ for $h=1,\dots,H$, and the total number of late jobs is u, where $0\leqslant y_h\leqslant q_h\leqslant n_h$, $0\leqslant v_h\leqslant q_h$ and $0\leqslant u\leqslant n$. If $v_h>0$ and $f(q,v,y,u)>d_{v_h,h}^M$ for any h, where $1\leqslant h\leqslant H$, then we define $f(q,v,y,u)=\infty$ and this replaces any assigned value. We define $v_h=0$ if there are no undelivered on-time jobs for customer $\mathscr{C}_h$, and $d_{0h}=\infty$.

Boundary Condition

$$f(0,\dots,0,0,\dots,0,0,\dots,0,0)=0.$$

Optimal Solution Value

$$\min\Big\{u+\sum_{h=1}^{H}D_h^M y_h\ \Big|\ f(n_1,\dots,n_H,0,\dots,0,y_1,\dots,y_H,u)<\infty, 0\leqslant u\leqslant n, 0\leqslant y_h\leqslant n_h \text{ for } h=1,\dots,H\Big\}.$$

Recurrence Relation

$$f(q,v,y,u)=\min\begin{cases}\min\limits_{h\in H_1}\{f(q',v,y,u-1)\}\\ \min\limits_{h\in H_2}\{p_{q_h,h}^M+\max\{f(q',v,y,u),r_{q_h,h}^M\}\}\\ \min\limits_{h\in H_3}\Big\{\min\limits_{v'_h\in V_1}\{p_{q_h,h}^M+\max\{f(q',v',y,u),r_{q_h,h}^M\}\}\Big\}\\ \min\limits_{h\in H_4}\Big\{\min\limits_{v'_h\in V_2}\{p_{q_h,h}^M+\max\{f(q',v',y',u),r_{q_h,h}^M\}\}\Big\}\end{cases}$$

where

$$q'=(q_1,\dots,q_{h-1},q_h-1,q_{h+1}\dots,q_H),$$
$$v'=(v_1,\dots,v_{h-1},v'_h,v_{h+1}\dots,v_H),$$

$y' = (y_1, \ldots, y_{h-1}, y_h - 1, y_{h+1} \ldots, y_H)$,
$H_1 = \{h \mid 1 \leqslant h \leqslant H, 0 \leqslant v_h < q_h\}$,
$H_2 = \{h \mid 1 \leqslant h \leqslant H, 0 < v_h < q_h, d^M_{v_h,h} \leqslant d^M_{q_h,h}\}$,
$H_3 = \{h \mid 1 \leqslant h \leqslant H, v_h = q_h > 0\}$,
$H_4 = \{h \mid 1 \leqslant h \leqslant H, v_h = 0, q_h > 0\}$,
$V_1 = \{v'_h \mid v'_h = 0$, or $0 < v'_h < q_h$ and $d^M_{v'_h,h} > d^M_{q_h,h}\}$, and
$V_2 = \{v'_h \mid 0 \leqslant v'_h < q_h,\ \max\{f(q', v', y', u),\ r^M_{q_h,h}\} + p^M_{q_h,h} \leqslant \min\{d^M_{v'_h,h},\ d^M_{q_h,h}\}\}$.

In the recurrence relation, the first term in the minimization schedules job (q_h, h) to be late, whereas the other three terms schedule job (q_h, h) on time. In the second term, no delivery is scheduled when job (q_h, h) completes processing, and job (v_h, h) remains the undelivered job with the smallest due date for customer $\mathscr{C}_h$. The third term also does not schedule a delivery when job (q_h, h) completes processing, but as specified by V_1, (q_h, h) is either the only undelivered on-time job for customer $\mathscr{C}_h$ or it has a smaller due date than the job (v'_h, h) which has the smallest due date among undelivered on-time jobs for customer $\mathscr{C}_h$ in the previous schedule. Thus, the third term is used only when $v_h = q_h$. The fourth term schedules a delivery when job (q_h, h) completes processing, and therefore is applied only when $v_h = 0$. The condition specified in V_2 ensures that all jobs of the batch that is scheduled for delivery are on time.

THEOREM 9. *Algorithm MU finds an optimal EDD-batch consistent schedule of on-time jobs for problem* $1|r_j|\sum U_j^M + \sum D_h^M y_h^M$ *in* $O(n^{3H+1})$ *time.*

PROOF. The dynamic program exploits properties of an optimal schedule from part (b) of Lemma 1, part (a) of Lemma 2, and the assumption of EDD-batch consistency for on-time jobs. There are $O(n^{3H+1})$ states (q, v, y, u). The first two terms in the recurrence relation require constant time. The third term requires $O(n)$ time, but is only applied for the $O(n^{3H})$ states where $v_h = q_h$ for some h. Similarly, the fourth term requires $O(n)$ time, but is only applied for the $O(n^{3H})$ states where $v_h = 0$ for some h. Therefore, the overall time complexity of Algorithm MU is $O(n^{3H+1})$. □

We now consider the corresponding weighted problem $1|r_j|\sum w_j U_j^M + \sum D_h^M y_h^M$.

THEOREM 10. *An optimal EDD-batch consistent schedule of on-time jobs for problem* $1|r_j|\sum w_j U_j^M + \sum D_h^M y_h^M$ *can be found in* $O(n^{3H}W)$ *time. The recognition version of this problem, under the assumption of EDD-batch consistency for the on-time jobs, is binary NP-complete.*

PROOF. We modify Algorithm MU by replacing the number of late jobs state variable u by the total weight of late jobs. The original time complexity of $O(n^{3H+1})$ in MU becomes $O(n^{3H}W)$ with this modification. The second result follows from the binary *NP*-completeness of the recognition version of the classical problem $1\|\sum w_j U_j$ (Karp 1972) and Theorem 1, where the latter is valid under the assumption of EDD-batch consistency for the on-time jobs. □

5. THE COMBINED PROBLEM

Here we assume that the supplier $\mathscr{S}$ and manufacturer $\mathscr{M}_1$ cooperate to solve a combined problem of minimizing the total system cost. There are single-stage jobs which the supplier $\mathscr{S}$ produces for manufacturers $\mathscr{M}_2, \ldots, \mathscr{M}_G$ with indexes (j, g), for $g = H+1, \ldots, H+G-1$ and $j = 1, \ldots, n_g$. There are also two-stage jobs for which parts are produced by the supplier and the final product is produced by manufacturer $\mathscr{M}_1$ for customers $\mathscr{C}_1, \ldots, \mathscr{C}_H$ with indexes (j, h), for $h = 1, \ldots, H$ and $j = 1, \ldots, n_h$. We refer to jobs $(1, k), \ldots, (n_k, k)$ for $k = 1, \ldots, H+G-1$ as *group* k. Thus, all the jobs in group k are destined for customer $\mathscr{C}_k$ if $1 \leqslant k \leqslant H$, and for manufacturer $\mathscr{M}_{k-H+1}$ if $H+1 \leqslant k \leqslant H+G-1$. We need to determine schedules for machines M_S and M_M, and to specify a batch delivery schedule from the supplier $\mathscr{S}$ to all manufacturers and from manufacturer $\mathscr{M}_1$ to all customers. The scheduling cost of the two-stage jobs is based on delivery times from $\mathscr{M}_1$ to the customers. This combined problem is a two-machine flow shop with batch deliveries, and with some missing operations at the second stage.

Because the recognition versions of the classical problems $F2\|\sum F_j$, $F2\|L_{\max}$, and $F2\|\sum U_j$ are unary *NP*-complete (Garey et al. 1976, Lenstra et al. 1977) and from Theorem 1, the recognition versions of the combined problems $1\|\sum F_j^C + \sum D_g^S y_g^S + \sum D_h^M y_h^M$, $1\|L_{\max}^C + \sum D_g^S y_g^S + \sum D_h^M y_h^M$, and $1\|\sum U_j^C + \sum D_g^S y_g^S + \sum D_h^M y_h^M$, respectively, are also unary *NP*-complete. Therefore, we first describe and then motivate a simplifying assumption for each of these problems. For the sum of flow times problem, we make the natural assumption that the single-stage jobs for each of the manufacturers $\mathscr{M}_2, \ldots, \mathscr{M}_G$ are sequenced in SPT order by the supplier according to the processing time on machine M_S, and that the two-stage jobs for each customer are sequenced by both the supplier and manufacturer $\mathscr{M}_1$ in SPT order according to the total processing time on machines M_S and M_M. This enables us to develop a polynomial time dynamic programming algorithm for that problem. We refer to this assumption as *total SPT within groups*. Similarly, by assuming that the single-stage jobs for each of the manufacturers $\mathscr{M}_2, \ldots, \mathscr{M}_G$ are sequenced in EDD order by the supplier, and that the two-stage jobs for each customer are sequenced by both the supplier and manufacturer $\mathscr{M}_1$ in EDD order, we develop a polynomial time algorithm for the maximum lateness problem. We refer to this assumption as *EDD within groups*. Finally, by making a similar *EDD within groups for the on-time jobs* assumption for the number of late jobs problem, we provide a pseudopolynomial time algorithm. Each of these three assumptions is motivated by a heuristic scheduling rule for the particular objective being considered. Note

that all these assumptions produce a permutation schedule, as defined in Lemma 3.

Let τ denote the set of all possible completion times on machine M_M. The following result shows that τ contains only a polynomial number of values.

LEMMA 7. *For problem* $1\|\gamma^C + \sum D_g^S y_g^S + \sum D_h^M y_h^M$, *where* $\gamma^C \in \{\sum F_j^C, L_{\max}^C\}$, *assuming a given ordering of the jobs within each group, we have* $|\tau| = O(n^{G+3H-1})$.

PROOF. From the given ordering, each partial schedule defines the number of jobs q_k of each group k for $k = 1, \ldots, H+G-1$ that are processed, where $0 \leqslant q_k \leqslant n_k$. From part (a) of Lemma 1, the time at which a job completes processing on machine M_S must be chosen from $O(n^{G+H-1})$ possible values. These values may be regarded as release dates of jobs for manufacturer $\mathcal{M}_1$. The time at which a job completes processing on machine M_M is some release date plus the total processing time on M_M of some jobs $(\bar{q}'_h+1, h), \ldots, (\bar{q}_h, h)$ for $h = 1, \ldots, H$, where $0 \leqslant \bar{q}'_h < \bar{q}_h \leqslant n_h$. There are $O(n^{2H})$ possible values of this total processing time. Therefore, considering all possible release dates and subsequent processing times, the maximum number of completion times on machine M_M is $O(n^{G+3H-1})$. □

5.1. Sum of Flow Times

In this subsection, we adopt the total SPT within groups assumption for problem $1\|\sum F_j^C + \sum D_g^S y_g^S + \sum D_h^M y_h^M$. Thus, we index the jobs such that $p_{1h}^S + p_{1h}^M \leqslant \cdots \leqslant p_{n_h,h}^S + p_{n_h,h}^M$ for $h = 1, \ldots, H$, and $p_{1g}^S \leqslant \cdots \leqslant p_{n_g,g}^S$ for $g = H+1, \ldots, H+G-1$.

Our dynamic programming algorithm combines the enumeration schemes used in Algorithms SF and MF. More precisely, we schedule batches of jobs on machine M_S and individual jobs on machine M_M. We define a state variable t which denotes the completion time of the current partial schedule on machine M_M. When scheduling a job on machine M_M, we ensure that it does not start before time T, the completion time of the current partial schedule on M_S.

Algorithm CF

Value Function

$f(q, \bar{q}, s, t) = f(q_1, \ldots, q_{H+G-1}, \bar{q}_1, \ldots, \bar{q}_H, s_1, \ldots, s_H, t)$ = the minimum total cost for the supplier processing jobs $(1, h), \ldots, (q_h, h)$ and delivering them to manufacturer $\mathcal{M}_1$ for $h = 1, \ldots, H$; for the supplier processing jobs $(1, H+g-1), \ldots, (q_{H+g-1}, H+g-1)$ and delivering them to manufacturer $\mathcal{M}_g$ for $g = 2, \ldots, G$; and for manufacturer $\mathcal{M}_1$ processing and delivering jobs $(1, h), \ldots, (s_h, h)$ to customer $\mathcal{C}_h$ for $h = 1, \ldots, H$, with jobs $(s_h+1, h), \ldots, (\bar{q}_h, h)$ for $h = 1, \ldots, H$ also processed by $\mathcal{M}_1$; and with the last job processed by manufacturer $\mathcal{M}_1$ completing at time t, where $0 \leqslant s_h \leqslant \bar{q}_h \leqslant q_h \leqslant n_h$, $0 \leqslant q_{H+g-1} \leqslant n_{H+g-1}$, and $t \in \tau$.

Boundary Condition

$$f(0, \ldots, 0, 0, \ldots, 0, 0, \ldots, 0, 0) = 0.$$

Optimal Solution Value

$$\min_{t\in\tau}\{f(n_1, \ldots, n_{H+G-1}, n_1, \ldots, n_H, n_1, \ldots, n_H, t)\}.$$

Recurrence Relation

$$f(q, \bar{q}, s, t) = \min \begin{cases} \min\limits_{q'\in Q'}\{f(q', \bar{q}, s, t)\} + D_1^S \\ \min\limits_{g\in G_1}\Big\{ \min\limits_{0\leqslant q''_g < q_g} \{(q_g - q''_g)T + D_{g-H+1}^S \\ \qquad\qquad + f(q'', \bar{q}, s, t)\}\Big\} \\ \min\limits_{h\in H_1}\{f^*(q, \bar{q}', s, t - p_{\bar{q}_h,h}^M)\} \\ \min\limits_{h\in H_2}\{f(q, \bar{q}', s, t - p_{\bar{q}_h,h}^M)\} \\ \min\limits_{h\in H_3}\Big\{ \min\limits_{0\leqslant s'_h < s_h} \{(\bar{q}_h - s'_h)t + D_h^M \\ \qquad + f^*(q, \bar{q}', s', t - p_{\bar{q}_h,h}^M)\}\Big\} \\ \min\limits_{h\in H_4}\Big\{ \min\limits_{0\leqslant s'_h < s_h} \{(\bar{q}_h - s'_h)t + D_h^M \\ \qquad + f(q, \bar{q}', s', t - p_{\bar{q}_h,h}^M)\}\Big\} \end{cases}$$

where

$f^*(q, \bar{q}, s, t) = \min_{0\leqslant t'\leqslant t}\{f(q, \bar{q}, s, t')\}$,
$q'' = (q_1, \ldots, q_{g-1}, q''_g, q_{g+1}, \ldots, q_{H+G-1})$,
$\bar{q}' = (\bar{q}_1, \ldots, \bar{q}_{h-1}, \bar{q}_h - 1, \bar{q}_{h+1}, \ldots, \bar{q}_H)$,
$s' = (s_1, \ldots, s_{h-1}, s'_h, s_{h+1}, \ldots, s_H)$,
$Q' = \{(q'_1, \ldots, q'_H, q_{H+1}, \ldots, q_{H+G-1}) \mid \bar{q}_h \leqslant q'_h \leqslant q_h$ for $1 \leqslant h \leqslant H$, $q'_h < q_h$ for some $h\}$,
$G_1 = \{g \mid H+1 \leqslant g \leqslant H+G-1, q_g > 0\}$,
$H_1 = \{h \mid 1 \leqslant h \leqslant H, s_h < \bar{q}_h, p_{\bar{q}_h,h}^M = t - T\}$,
$H_2 = \{h \mid 1 \leqslant h \leqslant H, s_h < \bar{q}_h, p_{\bar{q}_h,h}^M < t - T\}$,
$H_3 = \{h \mid 1 \leqslant h \leqslant H, s_h = \bar{q}_h > 0, p_{\bar{q}_h,h}^M = t - T\}$,
$H_4 = \{h \mid 1 \leqslant h \leqslant H, s_h = \bar{q}_h > 0, p_{\bar{q}_h,h}^M < t - T\}$, and
$T = \sum_{h=1}^{H+G-1} \sum_{j=1}^{q_h} p_{jh}^S$.

In the recurrence relation, the first term schedules a batch containing the two-stage jobs $(q'_h + 1, h), \ldots, (q_h, h)$ for $h = 1, \ldots, H$ to be processed on machine M_S, and this batch is delivered to manufacturer $\mathcal{M}_1$ on completion of its processing at time T. The condition $\bar{q}_h \leqslant q'_h$ in the definition of Q' ensures that jobs of this batch are not currently scheduled on machine M_M. At this stage, no scheduling cost is incurred for this batch of jobs. The second term schedules a batch $\{(q''_g + 1, g), \ldots, (q_g, g)\}$ of single-stage jobs, where $H+1 \leqslant g \leqslant H+G-1$, for processing on machine M_S and subsequent delivery to manufacturer $\mathcal{M}_{g-H+1}$ at time T, and evaluates the completion time and delivery cost of this batch accordingly. The third through sixth terms schedule the two-stage job $(\bar{q}_h, h)$ to be processed on machine M_M

in the interval $[t - p^M_{\bar{q}_h,h}, t]$ so that it completes at time t. In some cases, the time t' at which machine M_M completes processing the previous job in the partial schedule is such that $t' \leqslant T$, as in the third and fifth terms, or alternatively $t' = t - p^M_{\bar{q}_h,h} > T$, as in the fourth and sixth terms. In the third and fourth terms, $s_h < \bar{q}_h$, and thus no delivery of a batch containing job $(\bar{q}_h, h)$ to customer $\mathscr{C}_h$ occurs at time t. However, in the fifth and sixth terms, $s_h = \bar{q}_h$, and thus a delivery of batch $\{(s'_h + 1, h), \ldots, (\bar{q}_h, h)\}$ to customer $\mathscr{C}_h$ is scheduled at time t.

THEOREM 11. *Algorithm CF finds an optimal total SPT within groups schedule for problem* $1\|\sum F_j^C + \sum D_g^S y_g^S + \sum D_h^M y_h^M$ *in* $O(n^{2G+7H-2})$ *time.*

PROOF. The dynamic program exploits properties of an optimal schedule from part (a) of Lemma 1, Lemma 2, Lemma 3, and the total SPT within groups assumption. For any $(q, \bar{q}, s, t)$, where $t \geqslant 1$, $f^*(q, \bar{q}, s, t) = \min\{f^*(q, \bar{q}, s, t-1), f(q, \bar{q}, s, t)\}$ can be computed recursively in constant time. From Lemma 7, there are $O(n^{2G+6H-2})$ states $(q, \bar{q}, s, t)$. In the recurrence relation, the first term in the minimization requires $O(n^H)$ time, while the other terms require no more than $O(n)$ time. Therefore, the overall time complexity of Algorithm CF is $O(n^{2G+7H-2})$. □

5.2. Maximum Lateness

In this subsection, we adopt the EDD within groups assumption for problem $1\|L_{\max}^C + \sum D_g^S y_g^S + \sum D_h^M y_h^M$. Thus, we index the jobs such that $d^M_{1h} \leqslant \cdots \leqslant d^M_{n_h,h}$ for $h = 1, \ldots, H$, and $d^S_{1g} \leqslant \cdots \leqslant d^S_{n_g,g}$ for $g = H+1, \ldots, H+G-1$.

Our dynamic programming algorithm combines the enumeration schemes used in Algorithms SL and ML, and is similar to Algorithm CF.

Algorithm CL

Value Function

$f(q, \bar{q}, s, y, \bar{y}, t) = f(q_1, \ldots, q_{H+G-1}, \bar{q}_1, \ldots, \bar{q}_H, s_1, \ldots, s_H, y_1, \ldots, y_G, \bar{y}_1, \ldots, \bar{y}_H, t) =$ the minimum value of the maximum lateness for the supplier processing jobs $(1,h), \ldots, (q_h, h)$ and delivering them to manufacturer $\mathscr{M}_1$ using y_1 deliveries for $h = 1, \ldots, H$; for the supplier processing jobs $(1, H+g-1), \ldots, (q_{H+g-1}, H+g-1)$ and delivering them to manufacturer $\mathscr{M}_g$ using y_g deliveries for $g = 2, \ldots, G$; and for manufacturer $\mathscr{M}_1$ processing and delivering jobs $(1,h), \ldots, (s_h, h)$ to customer $\mathscr{C}_h$ using $\bar{y}_h$ deliveries for $h = 1, \ldots, H$, with jobs $(s_h+1, h), \ldots, (\bar{q}_h, h)$ for $h = 1, \ldots, H$ also processed by $\mathscr{M}_1$; and with the last job processed by manufacturer $\mathscr{M}_1$ completing at time t, where $0 \leqslant \bar{y}_h \leqslant s_h \leqslant \bar{q}_h \leqslant q_h \leqslant n_h$, $0 \leqslant y_1 \leqslant \sum_{h=1}^H q_h$, $0 \leqslant y_g \leqslant q_{H+g-1} \leqslant n_{H+g-1}$, and $t \in \tau$.

Boundary Condition

$$f(0, \ldots, 0, 0, \ldots, 0, 0, \ldots, 0, 0, \ldots, 0, 0, \ldots, 0, 0) = -\infty.$$

Optimal Solution Value

$$\min_{(y,\bar{y},t)\in Y} \left\{ f(n_1, \ldots, n_{H+G-1}, n_1, \ldots, n_H, n_1, \ldots, n_H, y, \bar{y}, t) + \sum_{g=1}^G D_g^S y_g + \sum_{h=1}^H D_h^M \bar{y}_h \right\},$$

where $(y, \bar{y}, t) = (y_1, \ldots, y_G, \bar{y}_1, \ldots, \bar{y}_H, t)$ and

$$Y = \left\{ (y, \bar{y}, t) \,\middle|\, 1 \leqslant y_1 \leqslant \sum_{h=1}^H n_h, 1 \leqslant y_g \leqslant n_{H+g-1} \text{ for } 2 \leqslant g \leqslant G, 1 \leqslant \bar{y}_h \leqslant n_h \text{ for } 1 \leqslant h \leqslant H, t \in \tau \right\}.$$

Recurrence Relation

$$f(q, \bar{q}, s, y, \bar{y}, t) = \min \begin{cases} \min\limits_{q' \in Q'} \{f(q', \bar{q}, s, y', \bar{y}, t)\} \\ \min\limits_{g \in G_1} \left\{ \min\limits_{1 \leqslant q''_g < q_g} \{\max\{T - d^S_{q''_g+1,g}, f(q'', \bar{q}, s, y'', \bar{y}, t)\}\} \right\} \\ \min\limits_{h \in H_1} \{f^*(q, \bar{q}', s, y, \bar{y}, t - p^M_{\bar{q}_h,h})\} \\ \min\limits_{h \in H_2} \{f(q, \bar{q}', s, y, \bar{y}, t - p^M_{\bar{q}_h,h})\} \\ \min\limits_{h \in H_3} \left\{ \min\limits_{0 \leqslant s'_h < \bar{s}_h} \{\max\{t - d^M_{s'_h+1,h}, f^*(q, \bar{q}', s', y, \bar{y}', t - p^M_{\bar{q}_h,h})\}\} \right\} \\ \min\limits_{h \in H_4} \left\{ \min\limits_{0 \leqslant s'_h < \bar{s}_h} \{\max\{t - d^M_{s'_h+1,h}, f(q, \bar{q}', s', y, \bar{y}', t - p^M_{\bar{q}_h,h})\}\} \right\} \end{cases}$$

where

$f^*(q, \bar{q}, s, y, \bar{y}, t) = \min_{0 \leqslant t' \leqslant t}\{f(q, \bar{q}, s, y, \bar{y}, t')\}$,
$y' = (y_1 - 1, y_2, \ldots, y_G)$,
$y'' = (y_1, \ldots, y_{g-H}, y_{g-H+1} - 1, y_{g-H+2}, \ldots, y_G)$,
$\bar{y}' = (\bar{y}_1, \ldots, \bar{y}_{h-1}, \bar{y}_h - 1, \bar{y}_{h+1}, \ldots, \bar{y}_H)$, and
q'', $\bar{q}'$, s', Q', G_1, H_1, H_2, H_3, H_4, and T are defined as in Algorithm CF.

The structure of the recurrence relation is similar to that in Algorithm CF.

THEOREM 12. *Algorithm CL finds an optimal EDD within groups schedule for problem* $1\|L_{\max}^C + \sum D_g^S y_g^S + \sum D_h^M y_h^M$ *in* $O(n^{3G+8H-2})$ *time.*

PROOF. The dynamic program exploits properties of an optimal schedule from part (a) of Lemma 1, Lemma 2, Lemma 3, and the EDD within groups assumption. From Lemma 7, there are $O(n^{3G+7H-2})$ states $(q, \bar{q}, s, y, \bar{y}, t)$. In the recurrence relation, the first term in the minimization requires $O(n^H)$ time, while the other terms require no more than $O(n)$ time. Therefore, the overall time complexity of Algorithm CL is $O(n^{3G+8H-2})$. □

5.3. Number of Late Jobs

Our first result shows that, assuming an EDD within groups sequence for the on-time jobs, problem $1\|\sum U_j^C +$

$\sum D_g^S y_g^S + \sum D_h^M y_h^M$ is intractable, even when there is only one manufacturer and one customer. A proof is provided by Hall and Potts (2000).

THEOREM 13. *The recognition version of problem* $1|\sum U_j^C + \sum D_g^S y_g^S + \sum D_h^M y_h^M$, *assuming an EDD within groups schedule for the on-time jobs, is binary NP-complete.*

Although Theorem 13 provides a negative result, we are still able to provide a reasonably practical algorithm for the more general problem $1\|\sum w_j U_j^C + \sum D_g^S y_g^S + \sum D_h^M y_h^M$, assuming an EDD within groups schedule for the on-time jobs. As in §5.2, we index the jobs such that $d_{1h}^M \leqslant \cdots \leqslant d_{n_h,h}^M$ for $h = 1, \ldots, H$, and $d_{1g}^S \leqslant \cdots \leqslant d_{n_g,g}^S$ for $g = H+1, \ldots, H+G-1$.

Our dynamic programming algorithm schedules one job at a time. To prevent a job being processed on M_M before it is delivered from the supplier, we assign a state variable to represent the time at which the current supplier's batch is to be delivered, and only schedule any further processing on machine M_M after this time.

Algorithm CU

Value Function

$f(q, v, t, z, \bar{t}, k)$ = $f(q_1, \ldots, q_{H+G-1}, v_1, \ldots, v_H, t, z, \bar{t}, k)$ = the total cost of processing the on-time jobs from $(1, l), \ldots, (q_l, l)$ on machine M_S and delivering them to the relevant manufacturer for $l = 1, \ldots, H+G-1$, and of processing the on-time jobs from $(1, h), \ldots, (q_h, h)$ on machine M_M and delivering those jobs except $(v_h, h), \ldots, (q_h, h)$ if $v_h > 0$ to customer $\mathscr{C}_h$ for $h = 1, \ldots, H$, given that machine M_S completes processing the on-time jobs at time t, the delivery of the current batch from M_S to M_M is scheduled at time z, machine M_M completes processing the on-time jobs at time $\bar{t}$, and k is the group of the last on-time job on machine M_S if $k > 0$, where $0 \leqslant v_h \leqslant q_h \leqslant n_h$ for $h = 1, \ldots, H$, $0 \leqslant q_g \leqslant n_g$ for $g = H+1, \ldots, H+G-1$, $0 \leqslant t \leqslant z \leqslant \bar{t} \leqslant \sum_{h=1}^{H+G-1} \sum_{j=1}^{n_h} p_{jh}^S + \sum_{h=1}^{H} \sum_{j=1}^{n_h} p_{jh}^M$, $z \leqslant \sum_{h=1}^{H+G-1} \sum_{j=1}^{n_h} p_{jh}^S$, and $0 \leqslant k \leqslant H+G-1$. If there are no undelivered on-time jobs for customer $\mathscr{C}_h$, then we set $v_h = 0$; otherwise, (v_h, h) is the undelivered job with the smallest due date.

Boundary Condition

$f(0, \ldots, 0, 0, \ldots, 0, 0, 0, 0, 0) = 0.$

Optimal Solution Value

$$\min_{(t,z,\bar{t},k)\in Y} \{f(n_1, \ldots, n_{H+G-1}, 0, \ldots, 0, t, z, \bar{t}, k)\},$$

where $Y = \{(t, z, \bar{t}, k) \mid 0 \leqslant t \leqslant z \leqslant P^S,\ 0 \leqslant z \leqslant \bar{t} \leqslant P^S + P^M,\ 0 \leqslant k \leqslant H+G-1\}$, $P^S = \sum_{h=1}^{H+G-1} \sum_{j=1}^{n_h} p_{jh}^S$ and $P^M = \sum_{h=1}^{H} \sum_{j=1}^{n_h} p_{jh}^M$.

Recurrence Relation

$$f(q, v, t, z, \bar{t}, k) = \begin{cases} \min\limits_{1\leqslant k'\leqslant H} \{w_{q_{k'},k'}^M + f(q', v, t, z, \bar{t}, k)\}, & \text{if } k = 0 \\ \min\limits_{H<k'\leqslant H+G-1} \{w_{q_{k'},k'}^S + f(q', v, t, z, \bar{t}, k)\}, & \\ \quad \text{if } k = 0 & \\ w_{q_k,k}^M + f(q'', v, t, z, \bar{t}, k), & \\ \quad \text{if } 1 \leqslant k \leqslant H,\ q_k > v_k & \\ w_{q_k,k}^S + f(q'', v, t, z, \bar{t}, k), & \\ \quad \text{if } H < k \leqslant H+G-1 & \\ f^*(q'', v, t - p_{q_k,k}^S, z, \bar{t} - p_{q_k,k}^M), & \\ \quad \text{if } k \in K_1,\ q_k > v_k > 0 & \\ f^*(q'', v', t - p_{q_k,k}^S, z, \bar{t} - p_{q_k,k}^M), & \\ \quad \text{if } k \in K_1,\ q_k = v_k > 0 & \\ \min\limits_{v_k''\in V_1} \{f^*(q'', v'', t - p_{q_k,k}^S, z, \bar{t} - p_{q_k,k}^M)\} + D_k^M, & \\ \quad \text{if } k \in K_1,\ v_k = 0 & \\ f^*(q'', v, t - p_{q_k,k}^S, t - p_{q_k,k}^S, \bar{t} - p_{q_k,k}^M) + D_1^S, & \\ \quad \text{if } k \in K_1,\ q_k > v_k > 0,\ \bar{t} - p_{q_k,k}^M > z & \\ \min\limits_{0\leqslant\bar{t}'\leqslant z} \{f^*(q'', v, t - p_{q_k,k}^S, t - p_{q_k,k}^S, \bar{t}')\} + D_1^S, & \\ \quad \text{if } k \in K_1,\ q_k > v_k > 0,\ \bar{t} - p_{q_k,k}^M = z & \\ f^*(q'', v', t - p_{q_k,k}^S, t - p_{q_k,k}^S, \bar{t} - p_{q_k,k}^M) + D_1^S, & \\ \quad \text{if } k \in K_1,\ q_k = v_k > 0,\ \bar{t} - p_{q_k,k}^M > z & \\ \min\limits_{0\leqslant\bar{t}'\leqslant z} \{f^{**}(q'', v', t - p_{q_k,k}^S, t - p_{q_k,k}^S, \bar{t}')\} + D_1^S, & \\ \quad \text{if } k \in K_1, q_k = v_k > 0, \bar{t} - p_{q_k,k}^M = z & \\ \min\limits_{v_k''\in V_1} \{f^*(q'', v'', t - p_{q_k,k}^S, t - p_{q_k,k}^S, \bar{t} - p_{q_k,k}^M)\} & \\ \quad + D_1^S + D_k^M, \quad \text{if } k \in K_1,\ v_k = 0,\ \bar{t} - p_{q_k,k}^M > z & \\ \min\limits_{v_k''\in V_1} \left\{ \min\limits_{0\leqslant\bar{t}'\leqslant z} \{f^{**}(q'', v'', t - p_{q_k,k}^S, t - p_{q_k,k}^S, \bar{t}')\} \right\} & \\ \quad + D_1^S + D_k^M, \quad \text{if } k \in K_1,\ v_k = 0,\ \bar{t} - p_{q_k,k}^M = z & \\ f(q'', v, t - p_{q_k,k}^S, z, \bar{t}, k), \quad \text{if } k \in K_2 & \\ \min\limits_{0\leqslant k'\leqslant H+G-1} \{f(q'', v, t - p_{q_k,k}^S, t - p_{q_k,k}^S, \bar{t}, k')\} & \\ \quad + D_{k-H+1}^S, \quad \text{if } k \in K_2 & \end{cases}$$

where

$f^*(q, v, t, z, \bar{t}) = \min_{1\leqslant k\leqslant H}\{f(q, v, t, z, \bar{t}, k)\}$,
$f^{**}(q, v, t, z, \bar{t}) = \min_{0\leqslant k\leqslant H}\{f(q, v, t, z, \bar{t}, k)\}$,
$q' = (q_1, \ldots, q_{k'-1}, q_{k'} - 1, q_{k'+1}, \ldots, q_{H+G-1})$,
$q'' = (q_1, \ldots, q_{k-1}, q_k - 1, q_{k+1}, \ldots, q_{H+G-1})$,
$v' = (v_1, \ldots, v_{k-1}, 0, v_{k+1}, \ldots, v_H)$,
$v'' = (v_1, \ldots, v_{k-1}, v_k'', v_{k+1}, \ldots, v_H)$,
$V_1 = \{v_k'' \mid 0 \leqslant v_k'' < q_k, v_k'' = 0 \text{ or } d_{k,v_k''}^M \geqslant \bar{t}\}$,
$K_1 = \{k \mid 1 \leqslant k \leqslant H,\ d_{q_k,k}^M \geqslant \bar{t}\}$, and
$K_2 = \{k \mid H < k \leqslant H+G-1, d_{q_k,k}^S \geqslant z\}$.

In the recurrence relation, there are 15 terms. The first two terms in the minimization schedule a job $(q_{k'}, k')$ to be late when no jobs are currently scheduled to be on time. In the first term, $(q_{k'}, k')$ is a two-stage job, whereas in the second term it is a single-stage job. The third and fourth terms schedule job (q_k, k) to be late when the schedule contains at least one on-time job. In the third term, the condition $q_k > v_k$ allows the two-stage job k to be late, whereas

the fourth term schedules a single-stage job to be late. The fifth through thirteenth terms schedule the two-stage job (q_k, k) in the interval $[t - p^S_{q_k,k}, t]$ on machine M_S and in the interval $[\bar{t} - p^M_{q_k,k}, \bar{t}]$ on machine M_M, so that it is on time. In the fifth, sixth, and seventh terms, job (q_k, k) does not start a new batch on machine M_S, and the previous job in the batch is of some group k', where $1 \leqslant k' \leqslant H$. In the fifth term, $v_k > 0$, thereby preventing a delivery to customer $\mathscr{C}_k$ when job (q_k, k) completes processing. The sixth term applies for $q_k = v_k > 0$ so that no delivery is scheduled on completion of job (q_k, k), but (q_k, k) is the first undelivered job in group k and the previous schedule has no undelivered job in group k. The seventh term has $v_k = 0$ so that a delivery to customer $\mathscr{C}_k$ is scheduled upon completion of job (q_k, k) on machine M_M at time $\bar{t}$. The condition on v''_k in the definition of V_1 in the seventh term ensures that all jobs in this batch are delivered to the customer by their due dates. The eighth through thirteenth terms correspond to the situation in which job (q_k, k) starts a new batch on machine M_S. The first two of these terms are analogous to the fifth term, the next two are analogous to the sixth term, and the last two are analogous to the seventh term. There are two terms in each case, depending upon whether job (q_k, k) starts processing on machine M_M at time z and is preceded by idle time in some interval $[\bar{t}', z]$, or job (q_k, k) is processed on machine M_M immediately after the previous job. The eleventh and thirteenth terms allow scheduling of the first on-time job. The fourteenth and fifteenth terms schedule the single-stage job (q_k, k), where $k > H$, in the interval $[t - p^S_{q_k,k}, t]$ on machine M_S so that it is on time. In the fourteenth term, this job is scheduled in the same batch as the previous job, whereas in the last term job (q_k, k) starts a new batch at time $t - p^S_{q_k,k}$ when the previous batch completes.

THEOREM 14. *Algorithm CU finds an optimal EDD within groups schedule for problem* $1 \| \sum w_j U^C_j + \sum D^S_g y^S_g + \sum D^M_h y^M_h$, *in* $O(n^{G+2H-1}P^3)$ *time.*

PROOF. The dynamic program exploits properties of an optimal schedule from part (a) of Lemma 1, Lemma 2, Lemma 3, and the EDD within groups for the on-time jobs assumption. There are $O(n^{G+2H-1}P^3)$ states $(q, v, t, z, \bar{t}, k)$. An analysis of the recurrence relation similar to that in the proof of Theorem 11 shows that the overall time complexity of Algorithm CU is $O(n^{G+2H-1}P^3)$. □

6. COOPERATION

We first show through three examples that substantial benefits can result from cooperation between the supplier $\mathscr{S}$ and manufacturer $\mathscr{M}_1$. Then, using two further examples, we discuss the stability of such cooperations and mechanisms for achieving them.

6.1. Benefits of Cooperation

We present an example for each of the three scheduling objectives in this paper. Each example shows that the solution which results from the supplier and manufacturer acting independently is considerably more costly than the solution of the combined problem.

EXAMPLE 1. Consider the following instance of the combined problem $1 \| \sum F^C_j + \sum D^S_g y^S_g + \sum D^M_h y^M_h$. There are two jobs $(n = 2)$, both produced by a single manufacturer $(G = 1)$ for a single customer $(H = 1)$, where the processing times and delivery costs are given in Table 2, and where K is an integer such that $K > 3$.

A straightforward cost comparison shows that there is an optimal schedule in which both the supplier and manufacturer process the jobs in the order $(1, 2)$ on machines M_S and M_M, respectively. For the supplier, processing of the two jobs is completed at times 1 and $K + 1$, respectively. Therefore, if the jobs are delivered in two separate batches, then the scheduling cost is $\sum F^S_j = 1 + (K + 1) = K + 2$. On the other hand, if the jobs are delivered in a single batch, then $\sum F^S_j = 2(K + 1)$. Thus, the overall costs $\sum F^S_j + \sum D^S_g y^S_g$ for two deliveries and one delivery are $(K + 2) + 2(K - 1) = 3K$ and $2(K + 1) + (K - 1) = 3K + 1$, respectively. Consequently, the supplier's decision is to deliver the two jobs in separate batches to arrive at the manufacturer at times 1 and $K + 1$, respectively.

For the manufacturer's problem, using release dates $r^M_1 = 1$ and $r^M_2 = K + 1$, processing of the two jobs is completed at times 2 and $K + 2$, respectively. Using two separate batches, the overall cost is $\sum F^M_j + \sum D^M_h y^M_h = 2K + 2$. Alternatively, using a single batch delivery, $\sum F^M_j + \sum D^M_h y^M_h = (K + 2) + K = 2K + 2$. Thus, the minimum total cost for the manufacturer's problem is $2K + 2$. To evaluate this schedule with respect to the combined problem, we add the supplier's optimal cost of $3K$, giving a total cost of $5K + 2$.

However, an optimal schedule for the combined problem is to deliver both jobs from the supplier to the manufacturer in a single batch at time $K + 1$. At time $K + 3$, after the manufacturer has processed both jobs, they are delivered to the customer in a single batch. The overall cost of this schedule is $\sum F^C_j + \sum D^S_g y^S_g + \sum D^M_h y^M_h = 2(K + 3) + (K - 1) + K = 4K + 5$. Therefore, the ratio of the cost when the supplier and manufacturer make separate decisions to the optimal cost is $(5K + 2)/(4K + 5)$, which can be arbitrarily close to 5/4 if K is large. Thus, cooperation provides a 20% reduction in total system cost.

EXAMPLE 2. Consider the following instance of the combined problem $1 \| L^C_{\max} + \sum D^S_g y^S_g + \sum D^M_h y^M_h$, where $n = 2$ and $G = H = 1$. The processing times, due dates, and delivery costs are given in Table 3, where K is an integer such that $K > 2$. The supplier's due dates are based on a schedule for the manufacturer in which job 1 is processed in

Table 2. Data for Example 1.

	Supplier		Manufacturer	
Job	1	2	1	2
Processing time	1	K	1	1
Delivery cost	$K - 1$		K	

Table 3. Data for Example 2.

	Supplier		Manufacturer	
Job	1	2	1	2
Processing time	1	K	1	1
Due date	1	$K+1$	2	$K+2$
Delivery cost	$K-1$		K	

the interval $[1, 2]$ and job 2 is processed in the interval $[K+1, K+2]$.

As in Example 1, it is easily verified that there exists an optimal schedule in which both the supplier and the manufacturer process the jobs in the order $(1, 2)$. The supplier chooses to deliver the jobs in two separate batches at times 1 and $K+1$ because the overall cost of $2K-2$ is less than $2K-1$, which is the cost that results from a single batch. Thus, $r_1^M = 1$ and $r_2^M = K+1$. For the manufacturer's problem, the overall cost is $2K$, either delivering the jobs in one or two batches. Evaluating this schedule with respect to the combined problem, we add the supplier's delivery cost of $2K-2$ to obtain a total cost of $4K-2$.

However, an optimal schedule for the combined problem is to deliver both jobs from the supplier to the manufacturer at time $K+1$, and to deliver both jobs from the manufacturer to the customer at time $K+3$, with an overall cost of $3K$. Therefore, the ratio of the cost when the supplier and manufacturer make separate decisions to the optimal cost is $(4K-2)/3K$, which can be arbitrarily close to 4/3 if K is large. Thus, cooperation provides a 25% reduction in total system cost.

EXAMPLE 3. Consider the following instance of the combined problem $1\|\sum U_j^C + \sum D_g^S y_g^S + \sum D_h^M y_h^M$, where $G = H = 1$. The processing times, due dates, and delivery costs are given in Table 4, where K is an integer and $K \geqslant n \geqslant 2$. The supplier's due dates are based on a schedule for the manufacturer in which jobs $1, \ldots, n-1$ are scheduled consecutively in the interval $[K-n+1, K]$ and job n is scheduled in the interval $[K, nK+1]$.

A unique optimal solution of the supplier's problem schedules job n to be on time, and all other jobs to be late and not delivered to the manufacturer. The manufacturer consequently schedules job n to be on time. Thus, the total cost is equal to $n-1$.

However, an optimal schedule for the combined problem schedules jobs $1, \ldots, n-1$ on time, where each job is delivered separately, and job n late, giving an optimal cost of 1. Thus, cooperation provides a reduction in total system cost that can be arbitrarily close to 100%.

Table 5. Data for Example 4.

	Supplier		Manufacturer	
Job	1	2	1	2
Processing time	1	K	1	1
Delivery cost	$K/2$		K	

The potential benefits of cooperation identified in Examples 1 through 3 can be compared with similar results from the supply chain management literature. For example, Parlar and Weng (1997) consider a problem of coordination between a firm's supply and manufacturing departments, and estimate the benefit of cooperation to total profit to be at least 26%.

6.2. Mechanisms for Cooperation

We now present two further examples for problem $1\|\sum F_j^C + \sum D_g^S y_g^S + \sum D_h^M y_h^M$. If the supplier's and manufacturer's schedules are used to form a combined schedule, then $F_j^C = F_j^S + F_j^M$. Thus, the overall cost in the combined problem is equal to the supplier's total cost plus the manufacturer's total cost.

EXAMPLE 4. Consider the following instance of problem $1\|\sum F_j^C + \sum D_g^S y_g^S + \sum D_h^M y_h^M$, where $n = 2$, $G = 1$, and $H = 1$. The processing times and delivery costs are given in Table 5, where K is an integer such that $K > 8$.

Some insights about cooperation can be gained from summarizing the results for Example 4 in the form of a payoff matrix, as in Table 6, where the notation (c_1, c_2): $c_1 + c_2$ denotes that the supplier's cost is c_1, the manufacturer's cost is c_2, and the total cost is $c_1 + c_2$. As in Example 1, the jobs are processed in the order $(1, 2)$ on machines M_S and M_M.

The solution which minimizes total cost in Table 6 requires both the supplier and the manufacturer to use a single batch. However, this solution is not a Nash equilibrium (Nash 1951, Myerson 1991), because the supplier can benefit by unilaterally deviating from it and using two batches. Mechanisms by which the manufacturer can obtain the supplier's cooperation include:

1. offering an incentive if all the jobs in the order are delivered together, to an extent sufficient to compensate the supplier for the increased cost; and

2. offering to share holding costs for completed jobs at the supplier.

A situation analogous to that in the first mechanism above arises in the personal computer industry, as discussed

Table 4. Data for Example 3.

	Supplier				Manufacturer			
Job	1	...	$n-1$	n	1	...	$n-1$	n
Processing time	K	...	K	$K-1$	1	...	1	$(n-1)K+1$
Due date	$K-n+1$	...	$K-1$	K	$(n-1)K+1$	...	$(n-1)K+1$	$nK+1$
Delivery cost	0				0			

Table 6. Payoff matrix for Example 4.

	Manufacturer	
Supplier	1 batch	2 batches
1 batch	$(5K/2+2, K+4)$: $7K/2+6$	$(5K/2+2, 2K+3)$: $9K/2+5$
2 batches	$(2K+2, 2K+2)$: $4K+4$	$(2K+2, 2K+2)$: $4K+4$

by Lee et al. (2000). Because of frequent declines in retail prices, retailers of personal computers tend to delay purchases, so as to avoid being caught with expensive inventory that can only be sold cheaply. Computer manufacturers may offer price protection packages to encourage earlier purchases.

Regarding the second mechanism above, our recommendations are similar to Propositions 1 and 2 of Moses and Seshadri (2000), where a necessary and sufficient condition for decision makers at two stages to agree on a stocking level is an agreement to share holding costs at the second stage. Also, Agrawal and Seshadri (2000) consider a model that determines the timing of replenishments from a supplier to multiple retail stores. Their discussion of the trade-off between immediate and delayed replenishment is analogous to our batching decision.

In the event that attempts at cooperation fail, the manufacturer can still attempt to force the supplier to deliver only one batch. Mechanisms for doing so include refusing to accept delivery before a certain date, or refusing to accept delivery of partial orders.

EXAMPLE 5. Consider the following instance of problem $1\|\sum F_j^C + \sum D_g^S y_g^S + \sum D_h^M y_h^M$ with $n = 2$, $G = 1$, and $H = 1$. The processing times and delivery costs are given in Table 7, where K is an integer such that $K \geqslant 2$.

The payoffs for Example 5 are summarized in Table 8, using the same format as Table 6, where the optimal processing order on machines M_S and M_M is $(1, 2)$. Here the solution that minimizes total cost requires the supplier to use two batches. Mechanisms by which the manufacturer can encourage the supplier to deliver the first job separately include:

1. offering an incentive for earlier delivery of part of the order, to an extent sufficient to compensate the supplier for the increased delivery cost;
2. offering to pick up part of the order (perhaps as part of an existing logistics operation); and
3. offering to share the extra delivery costs.

The linkages to the supply chain management literature discussed after Example 4 are again relevant here. Also, Agrawal and Seshadri (2000) address related timing issues in the sharing of inventory between a supplier and several manufacturers. If attempts at cooperation fail, then the manufacturer can still attempt to force the supplier to deliver a batch earlier, for example, by refusing to accept delivery of job 1 after a certain date.

Table 7. Data for Example 5.

	Supplier		Manufacturer	
Job	1	2	1	2
Processing time	1	K	K	K
Delivery cost	$3K/2$		K	

Table 8. Payoff matrix for Example 5.

	Manufacturer	
Supplier	1 batch	2 batches
1 batch	$(7K/2+2, 5K)$: $17K/2+2$	$(7K/2+2, 5K)$: $17K/2+2$
2 batches	$(4K+2, 4K)$: $8K+2$	$(4K+2, 4K)$: $8K+2$

In both Examples 4 and 5, the supplier and manufacturer can bargain over the savings in total system cost that are achieved by an optimal cooperative strategy. These savings are $(4K+4)-(7K/2+6) = K/2-2$ in Table 6, and $(17K/2+2)-(8K+2) = K/2$ in Table 8. The bargaining solution theory of Nash (1950, 1953) suggests that neither player will bargain at a cost greater than they can achieve independently. Thus, a solution with $c_1 \leqslant 2K+2$ and $c_2 \leqslant 2K+2$ is found for Example 4, and a solution with $c_1 \leqslant 7K/2+2$ and $c_2 \leqslant 5K$ is found for Example 5. The supply chain management literature offers some guidance about how the profits from cooperation should be divided. For example, Weng (1995) develops models which share the increased profit from cooperation. A similar methodology can be used here.

It is clear from Tables 6 and 8 that the supplier's cost is independent of the manufacturer's decisions, and therefore the supplier has no incentive to persuade the manufacturer to change behavior. For this reason, any cooperation must be initiated by the manufacturer. Although our analysis focuses on supply chains with three stages—namely a supplier, manufacturers, and customers—more generally there may be s stages, where stage s corresponds to the customers. In this case, cooperation should be initiated at stage $s-1$ by discussion with stage $s-2$. If this cooperation is successfully achieved, then stages $s-2$ and $s-1$ can jointly approach stage $s-3$ about cooperation. Continuing in this way, the negotiations work their way upstream until all parts of the supply chain are cooperating for maximum benefit.

7. CONCLUDING REMARKS

In this paper, we present several models for scheduling, batching, and delivery problems that arise in supply chains. In particular, we provide algorithms for scheduling jobs on machines and forming batches for delivery that can assist a supplier in minimizing the cost of meeting a manufacturer's requirements. Similar models and algorithms are developed for the problem faced by the manufacturer who supplies customers, where the manufacturer is constrained by the arrival times of parts in batches from the

supplier. We describe models and algorithms for the problem of minimizing the total system cost of a supplier and manufacturer who cooperate about scheduling and batching decisions. We also demonstrate that substantial total system cost reductions can be achieved by such cooperation, and suggest how this cooperation can be motivated and initiated. Therefore, our work has practical implications for the way in which scheduling and delivery decisions in supply chains are made.

A number of important issues remain open. In this paragraph, we discuss issues related to extending our work. First, the efficiency of our polynomial and pseudopolynomial time algorithms can be enhanced by the development of dominance rules and lower bounds. Second, several problems considered here are now known to be intractable, which motivates the need to design optimal enumerative, as well as heuristic, solution procedures. A third important research topic is the extension of our models to multistage supply chains. As we have demonstrated, some natural assumptions about the consistency of processing sequences are necessary to model such environments. Under such assumptions, multistage supply chain scheduling models can offer detailed guidance about the potential benefits of coordinated decision making. Fourth, there is a need to develop bargaining models that allow for imperfect or asymmetric information in the negotiations between a supplier and a manufacturer. Related to this are game-theoretic issues about the stability of possible cooperative agreements. Finally, our identification of the substantial benefits of cooperation and the potential difficulty of obtaining and enforcing such cooperation among independent decision makers, motivate the need for greater organizational integration in supply chains. For example, if a single company controls several stages within the chain, then cooperation should be easier to initiate and maintain, thereby leading to greater efficiency. Research is needed to evaluate the benefits and costs of such integration in various manufacturing environments.

Finally, we discuss more general issues for supply chain scheduling research. Our intention is to redirect supply chain management research away from strategic and stochastic models, and more towards tactical and deterministic models. This redirection offers several advantages. In tactical planning, the shorter time horizon reduces the amount of uncertainty in the data, thus providing better opportunities for accurately evaluating the benefits and costs of cooperative decision making. A similar comment applies to the consideration of deterministic models, relative to stochastic ones. Besides this redirection, there are several other general implications of our work for supply chain management research. First, we encourage the quantification of the potential benefits of cooperation through the construction of specific examples. Second, insights about cooperation seem to have similarities across different operational planning problems, an observation that may have useful consequences. Finally, and perhaps most importantly, we propose closer linkages with economic models, especially those involving game theory and imperfect information, to ensure the stability of profitable cooperations in supply chain management.

ACKNOWLEDGMENTS

This work is supported in part by the Summer Fellowship Program of the Fisher College of Business (The Ohio State University), by the National Science Foundation under grant DMI-9821033, by NATO Collaborative Research Grant CRG 950773, and by the CORMSIS visitor programme (University of Southampton). Helpful comments on an earlier draft were provided by Marc Posner (The Ohio State University), Mohammad Mazdeh (Brunel University, U.K.), and Kamal Shanazari (University of Liverpool, U.K.). An associate editor and two referees also made suggestions that improved the presentation of the results.

REFERENCES

Agrawal, V., S. Seshadri. 2000. Joint determination of allocation & the timing of inventories in a supply chain. Working paper, Stern School of Business, New York University, New York.

Ahmadi, J. H., R. H. Ahmadi, S. Dasu, C. S. Tang. 1992. Batching and scheduling jobs on batch and discrete processors. *Oper. Res.* **40** 750–763.

Albers, S., P. Brucker. 1993. The complexity of one-machine batching problems. *Discrete Appl. Math.* **47** 87–107.

Cheng, T. C. E., V. S. Gordon, M. Y. Kovalyov. 1996. Single machine scheduling with batch deliveries. *Eur. J. Oper. Res.* **94** 277–283.

Erengüç, Ş. S., N. C. Simpson, A. J. Vakharia. 1999. Integrated production/distribution planning in supply chains. *Eur. J. Oper. Res.* **115** 219–236.

Garey, M. R., D. S. Johnson. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, San Francisco, CA.

——, ——, R. Sethi. 1976. The complexity of flowshop and jobshop scheduling. *Math. Oper. Res.* **1** 117–129.

Graham, R. L., E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan. 1979. Optimization and approximation in deterministic machine scheduling: A survey. *Ann. Discrete Math.* **5** 287–326.

Hall, N. G., C. N. Potts. 2000. Supply chain scheduling: Batching and delivery. Working paper, Fisher College of Business, The Ohio State University, Columbus, OH. Revised October 2001.

——, M. A. Lesaoana, C. N. Potts. 2001. Scheduling with fixed delivery dates. *Oper. Res.* **49** 134–144.

Karp, R. M. 1972. Reducibility among combinatorial problems. R. E. Miller, J. W. Thatcher, eds. *Complexity of Computer Computations*. Plenum Press, New York, 85–103.

Lee, C.-Y., Z.-L. Chen. 2001. Machine scheduling with transportation considerations. *J. Scheduling* **4** 3–24.

Lee, H. L., V. Padmanabhan, T. A. Taylor, S. Wang. 2000. Price protection in the personal computer industry. *Management Sci.* **46** 467–482.

Lenstra, J. K., A. H. G. Rinnooy Kan, P. Brucker. 1977. Complexity of machine scheduling problems. *Ann. Discrete Math.* **1** 343–362.

Moses, M., S. Seshadri. 2000. Policy mechanisms for supply chain coordination. *IIE Trans.* **32** 245–262.

Myerson, R. B. 1991. *Game Theory: Analysis of Conflict*. Harvard University Press, Cambridge, MA.

Nash, J. F. 1950. The bargaining problem. *Econometrica* **18** 155–162.

——. 1951. Noncooperative games. *Ann. Math.* **54** 289–295.

——. 1953. Two-person cooperative games. *Econometrica* **21** 128–140.

Ow, P. S., S. F. Smith, R. Howie. 1988. A cooperative scheduling system. M. D. Oliff, ed. *Expert Systems and Intelligent Manufacturing*. Elsevier Science Publishing, Amsterdam, The Netherlands, 43–56.

Parlar, M., Z. K. Weng. 1997. Designing a firm's coordinated manufacturing and supply decisions with short product life cycles. *Management Sci.* **43** 1329–1344.

Potts, C. N., M. Y. Kovalyov. 2000. Scheduling with batching: A review. *Eur. J. Oper. Res.* **120** 228–249.

——, L. N. Van Wassenhove. 1992. Integrating scheduling with batching and lot-sizing: A review of algorithms and complexity. *J. Oper. Res. Soc.* **43** 395–406.

Sarmiento, A. M., R. Nagi. 1999. A review of integrated analysis of production-distribution systems. *IIE Trans.* **31** 1061–1074.

Thomas, D. J., P. M. Griffin. 1996. Coordinated supply chain management. *Eur. J. Oper. Res.* **94** 1–15.

Webster, S., K. R. Baker. 1995. Scheduling groups of jobs on a single machine. *Oper. Res.* **43** 692–703.

Weng, Z. K. 1995. Channel coordination and quantity discounts. *Management Sci.* **41** 1509–1522.

——. 1997. Pricing and ordering strategies in manufacturing and distribution alliances. *IIE Trans.* **29** 681–692.

Yang, X. 2000. Scheduling with generalized batch delivery dates and earliness penalties. *IIE Trans.* **32** 735–741.